Esa Tuononen

ASP.NET-TEKNIIKAN HYÖDYNTÄMINEN DYNAAMISTEN INTERNET-SIVUJEN TOTEUTTAMISESSA

Opinnäytetyö
Kajaanin ammattikorkeakoulu
Trademikoulutus
Syksy 2007

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Luonnontieteiden ala	Koulutusohjelma Tietojenkäsittelyn koulutusohjelma
Tekijä(t) Esa Tuononen	
Työn nimi ASP.NET-tekniikan hyödyntäminen dynaamisten internet-sivujen toteuttamisessa	
Vaihtoehtoiset ammattiopinnot Ohjelmointi	Ohjaaja(t) Raimo Mustonen
	Toimeksiantaja Kainuun Kissanystävät ry
Aika 15.11.2007	Sivumäärä ja liitteet 44+7

Opinnäytetyön tarkoituksena oli luoda dynaaminen Internet-sivusto yleishyödyllisen yhdistyksen toiminnan työkaluksi. Tavoitteena oli saada yhdistyksen käyttöön nykyistä paremmat työkalut tiedon keräämiseen, säilyttämiseen ja välittämiseen.

Teoriaosuudessa käsiteltiin Internet-sivuston suunnittelua ja käytettävyyttä sekä ASP.NET –tekniikan perusominaisuuksia. ASP.NET –tekniikkaa koskevaan teoriaosuuteen sisällytettiin muun muassa tietoturvaan ja navigointiin liittyviä ominaisuuksia.

Toimeksiantajana toimineelle Kainuun Kissanystävät ry:lle luotiin Internet-sivusto ASP.NET –tekniikalla ja sivustolla oleva tietokanta tehtiin käyttämällä Microsoft SQL Server Express Edition –sovellusta. Sivuston laatiminen painottui yhdistyksen käyttöön tarvittavien rekistereiden luomiseen, hallintaan ja päivittämiseen.

Yhdistyksen Internet-sivuston kehittämistä jatketaan opinnäytetyön tekemisen jälkeen, kun koko sivuston päivittäminen muutetaan selainpohjaiseksi.

Kieli	suomi
Asiasanat	ASP.NET 2.0, dynaaminen Internet-sivusto
Säilytyspaikka	☒ Kajaanin ammattikorkeakoulun Kaktus-tietokanta ☒ Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School Business	Degree Programme Data Processing
Author(s) Esa Tuononen	
Title Exploitation of ASP.NET Technique in Implementation of Dynamic Internet Applications	
Optional Professional Studies Programming	Instructor(s) Raimo Mustonen
	Commissioned by Kainuun Kissanystävät ry
Date 15 November 2007	Total Number of Pages and Appendices 44+7

The objective of the study was to create a dynamic internet application utilising the ASP.NET technique and a database by Microsoft SQL Server Express Edition. Another aim of the study commissioned by a non-profit association was to create registers which are maintained by a browser to improve gathering, handling and storage of information.

The theory of the study consists of two parts. The first part deals with planning of internet applications and their usability. The second part handles the ASP.NET technique, including among other things, features related to information security and navigation.

The content of the registers and the internet application was considered in cooperation with the members of the association. In addition, the user interface was designed based on the former internet application of the association.

From now on, the development of the association's internet application continues by changing the maintenance of the whole internet application as browser based.

Language of Thesis	Finnish
Keywords	ASP.NET 2.0, dynamic internet application
Deposited at	☒ Kaktus Database at Kajaani University of Applied Sciences ☒ Library of Kajaani University of Applied Sciences

SYMBOLILUETTELO

API	Application Programming Interface, sovellusohjelmointirajapinta
ASP.NET	ASP.NET on osa Microsoftin kehittämää .NET -arkkitehtuuria
	ASP.NETin avulla voidaan luoda XML-pohjaisia web-palveluita ja internet-sivustoja
CGI	Common Gateway Interface, verkkosovellusten tekemiseen tarkoitettu tekniikka
CLR	Common Language Runtime, .NET -arkkitehtuurin ajonaikainen ympäristö
	Ympäristö huolehtii muun muassa muistinhallinnasta ja säikeiden hallinnasta. CLR mahdollistaa monien erilaisten ohjelmointikielien käytön .NET –ympäristössä. Yleisimpiä ohjelmointikieliä ovat C# ja Visual Basic.NET.
CSS	Cascading Style Sheets, merkintäjärjestelmä
	Merkintäjärjestelmällä voidaan antaa ohjeita web-sivujen ja muiden dokumenttien ulkoasusta.
HTML	HyperText Markup Language, kuvauskieli
	Kuvauskielellä voidaan kuvata www-sivujen rakennetta. HTML:llä voidaan merkitä tekstin rakenne eli sitä mikä osa esimerkiksi tekstistä on otsikkoa ja mikä leipätekstiä. Merkintä tehdään web-dokumenttiin elementeillä ja elementeissä olevilla määritteillä.
JSP	JavaServer Pages, Java-pohjainen dynaamisten web-sovellusten tekemiseen tarkoitettu tekniikka
MySQL	Ruotsalaisen MySQL AB:n kehittämä tietokantapalvelin.
	Tietokannasta on saatavissa ilmainen sekä maksullinen versio

PHP	Hypertext Preprocessor, ohjelmointikieli
	PHP:llä voidaan toteuttaa dynaamisia web-pohjaisia sovelluksia.
XML	Extensible Markup Language, merkkauskieli
	XML sisältää dataa sekä tietoa datasta. Yleensä XML -muotoista tietoa käytetään viestintämenetelmänä eri sovellusten välillä. Kehittäjien on myös mahdollista määritellä XML:n avulla oma merkkauskieli.

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena on ASP.NET-tekniikan hyödyntäminen dynaamisten Internet-sivujen toteuttamisessa. Työn tavoitteena on luoda sivusto yleishyödyllisen yhdistyksen toiminnan työkaluksi. Uuden sivuston avulla pyritään kehittämään yhdistyksen toimintaan olennaisesti liittyvää tiedonhallintaa.

Opinnäytetyön teoriaosuus painottuu ASP.NET-tekniikan –erityisesti ASP.NET 2.0 version perusominaisuuksiin. Lisäksi opinnäytetyössä käsitellään yleisesti Internet-sivustojen suunnittelua ja käytettävyyttä huomioiden muun muassa sivuston rakenteen.

Internet-sovellusten tekemiseen on olemassa useita tekniikoita, mutta opinnäytetyössä keskitytään vain ASP.NET –tekniikan avulla luotaviin sovelluksiin. ASP.NET –tekniikka helpottaa sivustojen tekemistä ja niiden päivittämistä.

Opinnäytetyön toimeksiantajana on Kainuun Kissanystävät ry. Yhdistyksen tarkoituksena on hoitaa ja uudelleen sijoittaa kodittomia kissoja. Erittäin tärkeä osa yhdistyksen toimintaa on tiedottaminen ja valistaminen pääosin yhdistyksen Internet-sivuston avulla.

Laajentuneen toiminnan vuoksi Kainuun Kissanystävät ry tarvitsee käyttöönsä nykyistä paremmat työkalut tiedon keräämiseen, säilyttämiseen ja välittämiseen. Lisäksi yhdistys tarvitsee paikan, johon kerätä tietoa välitetyistä ja löytyneistä kissoista sekä yhdistyksen jäsenistä. Yhdistys tarvitsee myös nopean ja katkeamattoman reitin lisätä ja muuttaa tietoja sivustollaan, jotta tieto hylätyistä kissoista saadaan lähes välittömästi ihmisten tietoon.

Yhdistykselle tehdään Internet-sivusto, jonka avulla hallituksen kaikki jäsenet voivat ylläpitää jäsen- ja kissarekistereitä. Samalla helpotetaan ja nopeutetaan sivuston päivittämistä. Sivuston avulla rekisterit saadaan ajantasaisiksi ja tiedonvälitys jakaantuu useamman henkilön vastuulle yhdistyksen toivomusten mukaisesti.

2 WEB-SIVUSTON SUUNNITTELU JA KÄYTETTÄVYYS

Sivustoja suunniteltaessa on mietittävä, mitä tarkoitusta varten sivusto tehdään – onko sivuston tekemisen tarkoituksena vain, että on nykypäivää ”olla sähköisessä muodossa” vai onko sivuston tarkoituksena myös palvella jollakin tavalla esimerkiksi yrityksen tai yhteisön asiakkaita tai sidosryhmiä. Nykyisin yritykset käyttävät Internet-sivuja paljon muun muassa yrityksen toiminnan ja tuotteiden mainostamiseen. Sivuston suunnittelun merkitys korostuu, kun sivustolle tulevan tiedon määrä kasvaa, kun taas esimerkiksi henkilökohtaisille sivuille riittävät selkeät ja yksinkertaiset sivut, joissa sisältö ei muutu tiheään. (Korpela, J. & Linjama, T. 2005, 48 – 49.)

Kohderyhmien merkitys korostuu suurempaa sivustoa suunniteltaessa, jolloin kannattaa miettiä kenelle sivusto on esisijaisesti kohdennettu. Jos esimerkiksi yrityksen yhtenä kohderyhmänä ovat tuotteiden jälleenmyyjät toisessa maanosassa, on toki hyödyllistä, että he saavat yrityksen tuotteista tietoa käyttämällään kielellä. Vaikka kohderyhmien rajaus on tärkeää, voi sivustoa käyttää kuka tahansa Internetin käyttäjä, jolloin on tärkeää, että sivustolle saapunut käyttäjä saa yleiskuvan sivuston omistajasta, tuotteista ja palveluista sekä toiminnasta. Yleensä nämä asiat kuvataan aloitussivulla, jonka tarkoitus on myös houkutella käyttäjiä tutkimaan sivustoa tarkemmin. (Korpela, J. & Linjama, T. 2005, 49 – 50, 355.)

Osa sivuston suunnittelua on myös sivuston ylläpidon suunnittelu. Suunnitelmalla pyritään jakamaan vastuuta sivuston päivittämisestä sekä saamaan tietoa siitä, mitä tietoja tulisi päivittää milloinkin. On huomioitava, etteivät työkalut, joilla päivittäminen tehdään, saa olla vaikeita ja työläitä käyttää, jottei sivuston päivittäminen vähenny ajan myötä. Jos esimerkiksi sivuston päivittäminen tapahtuu selaimella käytettävällä käyttö-

liittymällä, on sivuston suunnittelijan tai tekijän huolehdittava myös liittymän käytön opastuksesta. (Korpela, J. & Linjama, T. 2005, 56 – 57.)

Web-sivustolla liikkumisen perustana on navigointi, jonka avulla käyttäjää ohjataan ja opastetaan löytämään sivustolta haluamansa asiat. Sivustolla liikkuminen pitäisi rakentaa siten, että käyttäjällä on tiedossaan koko ajan millä sivulla hän on, mistä hän kyseiselle sivulle tuli ja minne sivulta pääsee. (Nielsen, J. 2000, 188.)

Jokaisesta sivusta pitäisi käydä ilmi, kenen tai minkä sivuston sivulla käyttäjä on kyseisenä hetkenä ja miten käyttäjä pääsee takaisin etusivulle. Yleensä sivustolle asetetaan logo tai vastaava tunnus, joka näkyy kaikilla sivulla ja toimii samalla linkkinä sivuston etusivulle. (Nielsen, J. 2000, 191; Wiio, A. 2004, 158 – 159.)

Alla olevassa kuvassa (Kuvio 1.) on ruudunkaappauskuva Kajaanin ammattikorkeakoulun keväällä 2007 käyttöön otetusta uudistuneesta sivustosta. Kuvasta käy hyvin ilmi, kuinka montaa erilaista navigointitapaa voi sivustolla käyttää samanaikaisesti.

Kuvio 1. Kajaanin ammattikorkeakoulun Internet-sivusto

Liikkumiseen sivustolla vaikutetaan myös sivuston rakenteella, joka on yleensä hierarkkinen silloin kun sivustolla on paljon erilaista tietoa. Rakenteen hierarkkisuudella mahdollistetaan, että käyttäjä löytää asiakokonaisuuteen liittyvät tiedot valitun linkin alta. Esimerkiksi ”Yhdistys”-valikon alta löytyy tietoa toiminnasta, yhdistyksen historiasta jne. Tällöin käyttäjälle luodaan mielikuva yhdellä sanalla siitä mistä hänen etsimänsä tieto löytyy. (Nielsen, J. 2000, 198, 202.)

Käyttäjille näkyvin osa on sivuston visuaalinen ilme, jolla vaikutetaan myös sivuston ”suosioon”. Jos sivustolla on paljon välkkyviä mainoksia ja aiheeseen kuulumattomia osia, jotka vievät suurimman osan sivun tilasta, ei käyttäjä ole halukas tutkimaan sivustoa enempää eikä palaamaan sivustolle toiste. Sivuston yhtenäisellä ulkoasulla, esimerkiksi sisällön samankaltaisella esittämisellä kaikilla sivuilla, mahdollistetaan, että käyttäjä huomaa jo muuttuneesta ulkoasusta poistuneensa sivustolta ja menneensä uudelle sivustolle. (Korpela, J. & Linjama, T. 2005, 143, 356 – 357, 360.)

Dynaamisissa sivuissa olennainen osa käytettävyyttä on lomakkeiden helppokäyttöisyys ja vuorovaikutus käyttäjän ja sivuston välillä. Lomakkeiden täyttäminen tapahtuu vasemmalta oikealle (edellyttäen, että sivusto on suunniteltu länsimaiseen käyttöön) ja ylhäältä alas eli samaan tapaan kuin normaalisti on totuttu täyttämään paperilomakkeita. Sivuston tulisi myös kommunikoida käyttäjän kanssa eli ilmoittaa onnistuneesta tai epäonnistuneesta toimenpiteestä. Tärkeää on myös tarkistaa käyttäjän antamien syötteiden oikeellisuus. Syötteiden oikeellisuuden tarkistamisella pyritään varmistamaan, että väärät henkilöt eivät pääse suorittamaan omaa koodiaan esimerkiksi tietokantaan. (Kuutti, W. 2003, 56 – 62.)

3 ASP.NET 2.0

Dynaamisten selainsovellusten tekemiseen suunniteltu ASP.NET ei ole tarkoitettu ainoastaan Internet-sivujen tekemiseen, vaan ASP.NETllä voidaan tehdä myös XML-pohjaisia Web-palveluja. Muita web-sovellusten tekemiseen tarkoitettuja tekniikoita ovat esimerkiksi PHP ja CGI sekä Java-pohjaiset JSP (Java Server Pages) ja Servletit.

ASP.NET on osa Microsoftin kehittämää .NET-arkkitehtuuria. .NET Framework on ajonaikainen ympäristö, jonka avulla on mahdollista tehdä Windows- ja Web-sovelluksia. Tehdyt sovellukset suoritetaan .NET Frameworkin sisällä. Ajonaikainen ympäristö huolehtii muun muassa roskienkeruusta ja säikeiden hallinnasta. .NET Framework ympäristöön voidaan tehdä sovelluksia eri ohjelmointikielillä edellyttäen, että ne tukevat Common Language Runtimea (CLR). Tunnetuimpia ohjelmointikieliä .NET ympäristössä ovat C# ja Visual Basic.NET. (MSDN e.)

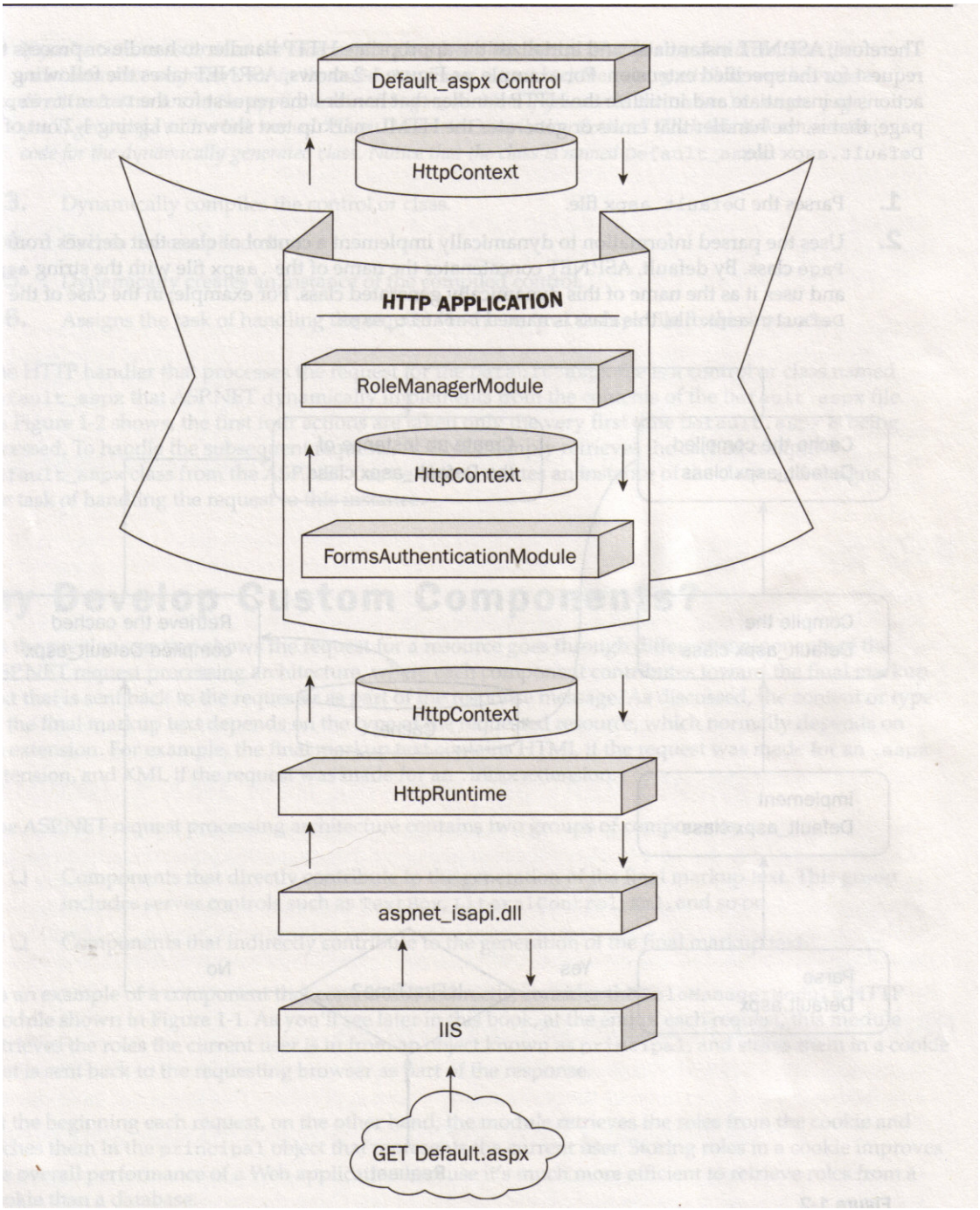
ASP.NETillä tehtävä Internet-sivu koostuu WebForm-lomakkeesta. Lomake koostuu yleensä direktiiveistä, joilla ohjataan sivun toimintaa, palvelimella suoritettavista koodilohkoista tai erillisestä luokkatiedostosta, joissa sijaitsee sivun varsinainen toiminnallisuus sekä HTML-osiosta, joka voi sisältää HTML-elementtejä, ASP.NET-palvelinkontrolleja tai ASP.NET-lohkoja. (Inkinen, V. 2003, 65.)

ASP.NETillä luotava sivusto voi sisältää useita eri tiedostoja ja yleisin on juuri Webform-lomakkeesta koostuva Internet-sivu. ASP.NETillä muodostettavan sivun tiedostopääte on .aspx, lisäksi muita sovelluksen käyttämiä tiedostoja ovat tiedostotyypiltään omat kontrollitiedostot (.ascx) sekä erilaiset sovelluksen konfigurointitiedostot (.config). Liitteessä 1 on lueteltu ASP.NETin käyttämät tiedostotyypit. (Inkinen, V. 2003, 61.)

3.1 Sivun muodostuminen Web-palvelimella

Käyttäjän pyytäessä ASP.NET tekniikalla luotua sivua web-palvelimelta, IIS (Internet Information Services) ottaa pyynnön vastaan ja lähettää sen aspnet_isapi.dll -nimiselle ISAPI laajennukselle. Seuraavana pyyntö lähetetään HttpRuntime nimisille .NET komponentille, joka luo instanssin HttpContext nimisestä .NET luokkasta. HttpContext sisältää tietoja pyynnöstä, josta se muodostaa objektit ”Request”, ”Response” jne. HttpContextista pyyntö etenee HttpApplication nimiselle .NET komponentille, joka luo yhteyden HTTP-moduuleille suorittaakseen esikäsitellyt operaatiot kuten tunnistautuminen ja käyttöoikeudet. (Khosravi, S. 2006, 2 – 3.)

Kun HTTP-moduulit ovat esikäsitelleet pyynnön, HttpApplication lähettää pyynnön HTTP- käsittelijäkomponentille (HTTP handler), jonka tehtävänä on tuottaa kuvauskieltä ja lähettää se osana vastausta pyynnön tekijälle. Erityyppiset HTTP-käsittelijät käsittelevät pyyntöjä eritavalla, esimerkiksi pyyntö, jonka on tarkoitus käsitellä .aspx laajennusta, tuottaa HTML-tekstiä vastaukseksi ja puolestaan käsittelijä, jonka tarkoituksena on käsitellä .asmx laajennusta tuottaa XML-kuvauskieltä vastaukseksi. Kun pyyntö on käsitelty ja muutettu oikeaan muotoon, se lähetetään takaisin pyytäjälle ja esitetään halutulla tavalla. (Kuvio 2.) (Khosravi, S. 2006, 2 – 3.)

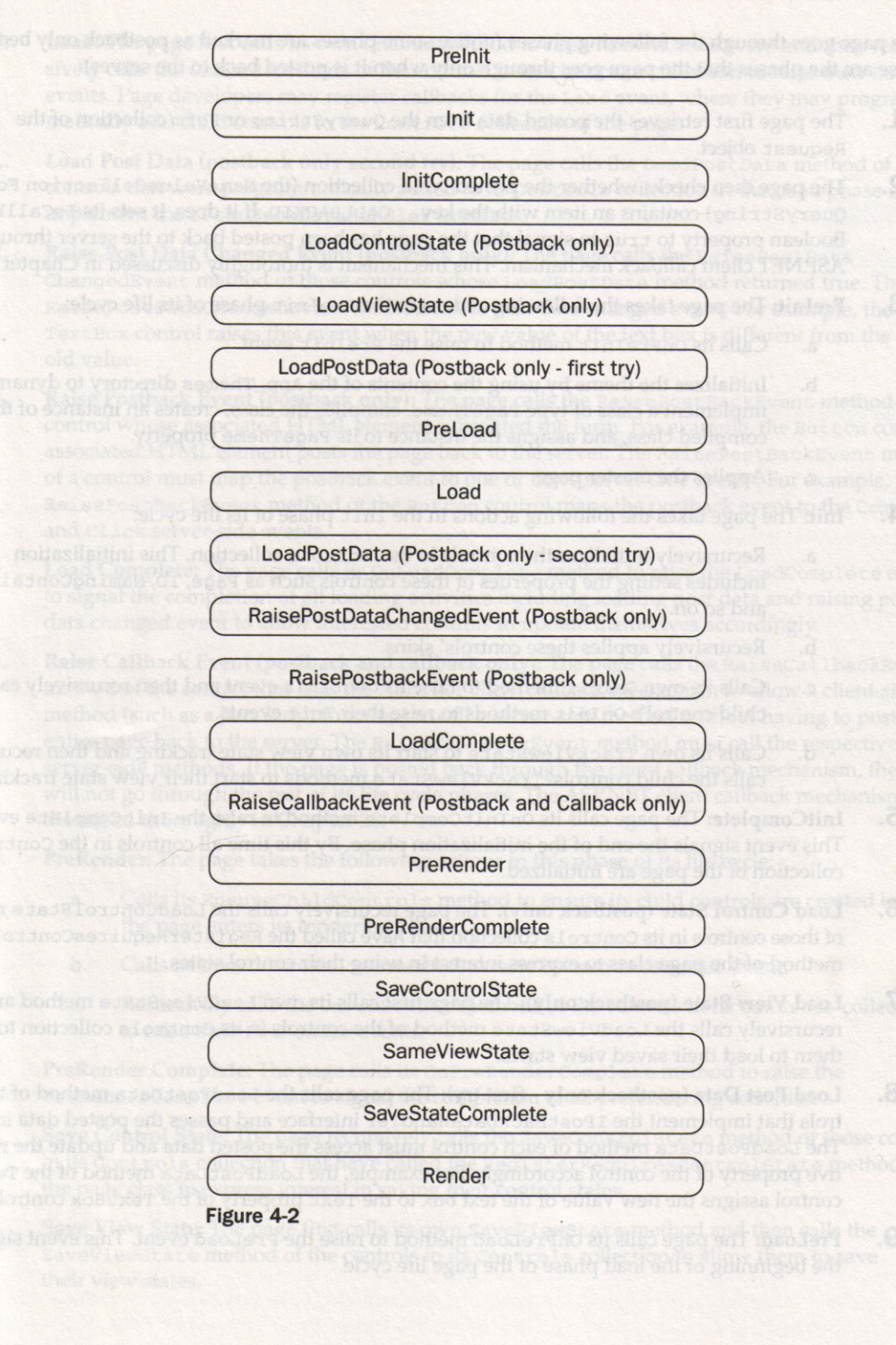


Kuvio 2. Sivun muodostumisprosessi (Khosravi, S. 2006, 3.)

3.2 Sivun elinkaari

Kun käyttäjä pyytää sivua palvelimelta, sivu muodostetaan vaiheittain tietyn kaavan mukaan. Kaikkia vaiheita ei suoriteta aina, vaan esimerkiksi lähetettäessä sivu uudelleen palvelimelle jätetään osa vaiheista suorittamatta. Alla olevassa kuviossa (Kuvio 3.) on

kuvattu sivun elinkaari pyynnön saapuessa palvelimelle ja sen rakentumisesta sivun pyytäneelle selaimelle. (Khosravi, S. 2006, 86 – 90.)



Kuvio 3. Sivun elinkaari. (Khosravi, S. 2006, 87.)

ASP.NETillä luodulla sivulla reagoidaan palvelinpäässä käyttäjän selaimessa tekemiin valintoihin ja pyyntöihin tapahtumilla, joista yleisimmät ovat sivun lataus ja esimerkiksi sivulla olevan napin painamisesta aiheutuva tapahtuma. Näillä tapahtumilla voidaan vaikuttaa sivulla esitettävään tietoon ja reagoida esimerkiksi käyttäjän tekemien valintojen oikeellisuuden tarkistamisella. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 79.)

3.3 Sovelluksen kansiot

Kehitettäessä Web-sovellusta Visual Studio 2005 työkaluilla, ASP.NET 2.0 käyttää tiedostopohjaista lähestymistä, joka tarkoittaa, että sovellukseen voi lisätä tiedostoja ja kansioita ilman, että sovellusta tarvitsee kääntää useaan kertaan. Aiemmassa ASP.NET versiossa ASP.NET käänsi kaiken sovelluksessa dll:ksi, 2.0 -versiossa tämä ei ole tarpeen, koska siihen on määritelty valmiiksi erikoiskansiorakenne. Kansiorakennetta hyväksikäyttämällä koodi automaattisesti käännetään ASP.NETin puolesta, ilman että sovelluksen kehittäjän tarvitsee tehdä sitä itse. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 88.)

Kansiorakenne sisältää muutamia erikoiskansioita, joilla jokaisella on oma tarkoituksensa. App_Code kansio on tarkoitettu sisältämään luokka- ja .wsdl-tiedostot sekä datasetit. Käytettäessä Visual Studio 2005 sovelluskehitintä, se huomaa automaattisesti kansioon lisätyn tiedoston ja kääntää sen. Kun sovelluskehitin on kääntänyt esimerkiksi luokan, on luokan tiedot käytettävissä saman tien millä tahansa sovelluksessa sijaitsevalla .aspx-sivulla. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 88.)

App_Code -kansioon lisättyjen luokkatiedostojen ei tarvitse olla samaa ohjelmointikieltä, vaan samassa sovelluksessa voidaan käyttää myös muita .NET ohjelmointikieliä. Eri ohjelmointikielillä tehtyjä luokkatiedostoja ei voi lisätä kuitenkaan suoraan App_Code -kansion juureen, vaan niille on luotava omat kansionsa, lisäksi Web.config -tiedostoon on lisättävä esimerkin mukainen määrittely.

Esimerkki:

<compilation>

< codeSubDirectoryName>

<add directoryName=”VB”> </add>

<add directoryName =”CS”> </add>

</ codeSubDirectoryName>

</compilation> (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 88 – 92.)

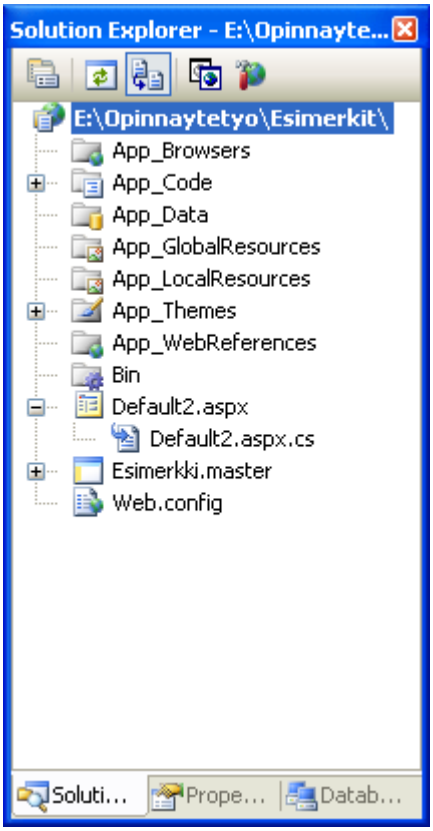
App_Data -kansio on tarkoitettu sovelluksen käyttämään tietojen varastoimiseen. Kansio voi sisältää niin Microsoft Sql Expressillä kuin Microsoft Accessilla luotuja tietokantoja ja XML-tiedostoja sekä paljon muita tietojen varastoimiseen käytettäviä tiedostoja. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 93.)

App_Theme -kansioon tallennetaan .skin- (kuoret) ja CSS-tiedostoja sekä kuvia, joiden voidaan luoda teemoja (Themes). Teemoilla voidaan vaikuttaa sivuston visualiseen ilmeeseen. Teemoista kerrotaan lisää luvussa 3.6.. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 93.)

Lähdetiedostoja (.resx), joilla voidaan esimerkiksi vaihtaa sivustolla käytettävä kieli samaksi kuin käyttäjän selaimessa on määritelty käytettäväksi, voidaan tallentaa App_GlobalResources kansioon. App_LocalResources -kansioon voidaan tallennetaan samanlaisia lähdetiedostoja kuin App_GlobalResources -kansioonkin. Ero näiden kansioiden sisältämillä tiedostoilla on, että App_GlobalResources -kansiossa olevilla tiedostoilla voidaan vaikuttaa koko sovelluksen alueella, kun taas App_LocalResources -kansioon lisätyt tiedostot on tarkoitettu käytettäväksi vain yhdelle .aspx-sivulle. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 93.)

App_WebReferences -kansio on tarkoitettu sivustolla käytettävien omien tai muiden tekemien Web-palveluiden sijaintikansioksi. Lisäksi sovellukseen voi lisätä App_Browsers nimisen kansion, joka sisältää .browsers tiedostoja. Näillä tiedostoilla voidaan hyödyntää sovelluksessa pyyntöä lähettävän selaimen erikoisominaisuuksia. Lisäksi on käytössä Bin- kansio joka sisältää sovelluksen käyttämät dll-tiedostot. Alla olevassa kuvassa

(Kuvio 4.) kansiorakenne on kuvattu Visual Web Developer 2005 Express Editionissa. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 94.)



Kuvio 4. Kansiorakenne

3.4 ASP.NET 2.0 palvelindirektiivit

Direktiivit ovat osa jokaista ASP.NET sivua ja kontrollia. Direktiivit ovat komentoja, joita kääntäjä käyttää sivua kääntäessään. Direktiivit sijoitetaan yleensä sivun tai komponentin yläosaan kehittäjien perinteiden mukaan. Direktiivit ovat muotoa <%@ [Direktiivi] [Atribuuti=Arvo] %>. Direktiivejä on käytössä useita erilaisia (Liite 2), joista kolme yleisintä ovat @Page, @Master ja @Contol. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 68.)

Esimerkissä on käytössä Page-direktiivi ja sen Language-attribuutti sekä sen arvo (C#).

Esimerkki:

```
<%@ Page Language="C#" %>
```

3.4.1 @Page

Direktiivi mahdollistaa attribuuttien ja arvojen määrittelyn ASP.NET sivuille (tiedostopääte .aspx) käytettäväksi kun sivua käännetään. Koska .aspx-sivut ovat olennainen osa ASP.NETiä, on attribuutteja käytettävissä varsin paljon. Alla olevassa esimerkissä @Page-direktiivissä on kolme attribuuttia: Language (käytettävä ohjelmointikieli), CodeFile (kooditiedosto) ja Inherits (periytyminen). (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 69 – 70.)

Esimerkki:

```
<%@ Page Language="C#" Theme="Theme" CodeFile="Default4.aspx.cs" Inherits="Default4" %>
```

Language-attribuutin arvolla määritellään käytettävä ohjelmointikieli, edellisessä esimerkissä C#. Theme-attribuutin arvo määrittelee sivulla käytettävän teeman, Teemoista kerrotaan lisää luvussa 3.6. CodeFile-attribuutin arvolla viitataan CodeBehind-tiedostoon, eli erilliseen luokkatiedostoon, johon voidaan tehdä kyseisen sivun toiminnallisuus. Käytettäessä erillistä luokkatiedostoa voidaan näin erityttää erillisiksi tiedostoiksi varsinainen käyttöliittymä ja sen toiminnallisuus. Inherits-atribuutin arvolla määritellään perittävä luokka. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 69 – 70.)

3.4.2 @Master

Direktiivi on samantapainen kuin @Page-direktiivi, mutta @Master-direktiivi on tarkoitettu MasterPage-tiedostoille (tiedostopääte .master). MasterPages:sta on kerrottu lisää luvussa 3.5. Vaikka @Page- ja @Master-direktiivit ovat samantapaisia, on @Master-direktiivillä käytettävissä vähemmän attribuutteja kuin @Page-direktiivilla. Alla olevassa

esimerkissä on käytetty samoja attribuutteja kuin esimerkissä koskien @Page-direktiiviä, mutta erona on Theme-attribuutin puuttuminen, muilta osin esimerkkien attribuutit tarkoittavat samaa. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 71 – 72.)

Esimerkki:

```
<%@ Master Language="C#" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
```

3.4.3 @Control

@Control-direktiivi on samantapainen kuin @Page-direktiivi, mutta @Control-direktiiviä käytetään luotaessa omia kontrolleja. Omista ja muista palvelinkontrolleista on kerrottu lisää luvussa 3.8. Alla olevassa esimerkissä on käytössä samat attribuutit kuin esimerkissä koskien @Master-direktiiviä. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 73 – 74.)

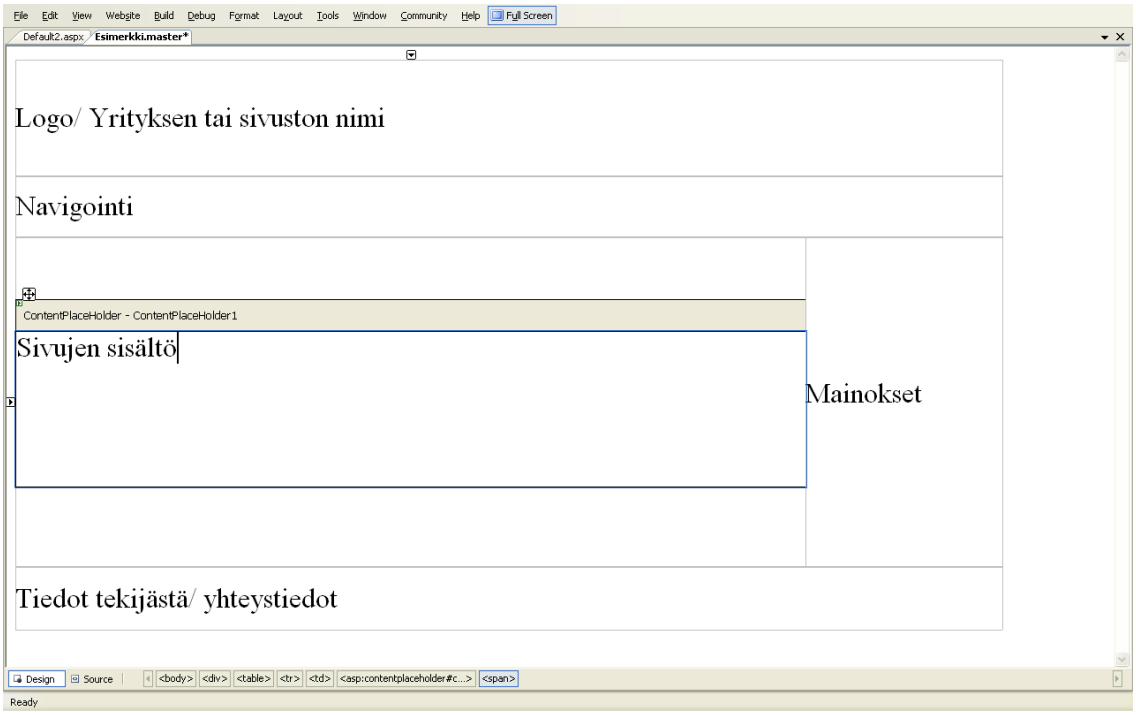
Esimerkki:

```
<%@ Control Language="C#" CodeFile="OmaKontrolli.ascx.cs" Inherits=" OmaKontrolli " %>
```

3.5 MasterPages

MasterPages ovat ”sivustopohjia”, joiden avulla koko sivustolle saadaan yhtenäinen ilme. MasterPages:ien avulla sivu voidaan jakaa eri osiin (navigointi, yläosa, jossa on logo tai sivuston nimi tms.). Sivun jakaminen mahdollistaa, ettei kaikkia osia tarvitse tehdä jokaiselle sivulle ja halutut osat näkyvät kuitenkin koko sivustolla. MasterPagen käyttäminen mahdollistaa myös, että sivuston muutokset tarvitsee tehdä vain yhteen paikkaan. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 277.)

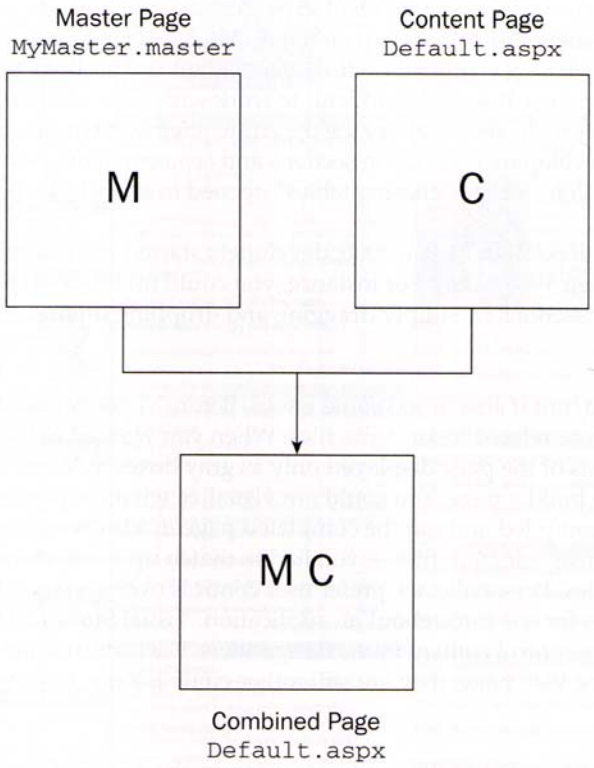
Alla olevassa kuvassa (Kuvio 5) on Visual Web Developer 2005 Express Editionilla tehty MasterPages -tiedosto (tiedostopääte on .master). MasterPages on jaettu viiteen eri osaan (Sivuston nimi, Navigointi, Mainokset, Tiedot sivuston tekijästä sekä Sivujen sisältö). ContentPlaceHolder-niminen palvelinkontrolli määrittelee alueen MasterPages:n sisällä, jossa .aspx sivut voivat näyttää tietonsa. MasterPages -tiedostot voivat sisältää samoja kontrolleja, kuvia, HTML-elementtejä ja tekstiä kuten .aspx-sivuilla normaalistikin. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 283.)



Kuvio 5. MasterPages

MasterPages muistuttaa hyvin paljon HTML:ssä käytettäviä kehyksiä (Frame), joilla luodaan yksi (yleensä index.html niminen) sivu, joka jaetaan osiin ja jokaiseen osaan ladataan oma HTML-päätteinen sivu (navigointi, sivuston tunnus jne.). Jaetuista osista yksi osa varataan muiden sisältösivujen esittämiseen. Kehyksiä käyttämällä mahdollistetaan sivustolle yhtenäinen ilme, tosin ongelmaksi voi muodostua, ettei sivupohja (index.html) välttämättä lataudu, kun käyttäjä tulee sivustolle muuta kautta (esimerkiksi hakukoneella tai suosikkien kautta) kuin sivuston suunnittelija oli ajatellut. Tällöin käyttäjälle näytetään vain kyseinen sisältösivu ilman, että käyttäjällä olisi esimerkiksi käytettävissään sivuston navigointi.

Suurin ero perinteisen HTML:ssä käytettävien kehyksien ja MasterPages:n välillä on, että MasterPages:n sisältösivuilla (.aspx tiedostopäätteiset) viitataan @Page-direktiivin MasterPageFile-attribuutin arvolla käytettävään MasterPages-tiedostoon (<%@Page MasterPageFile="~/Esimerkki.master" %>). Kun sivua ladataan palvelimelta ASP.NET yhdistää MasterPagen ja sisältösivun yhdeksi käyttäjälle näytettäväksi sivuksi kuvion mukaisesti (Kuvio 6.). (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 280.)

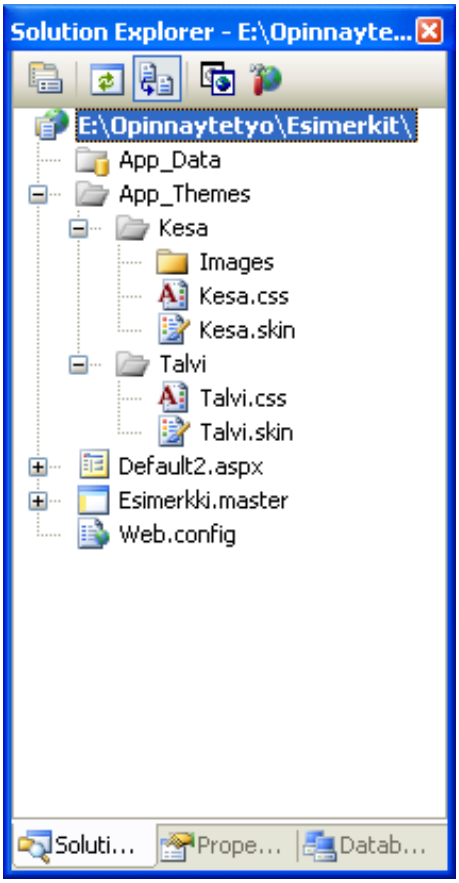


Kuvio 6. Yhdistyvät sivut (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 280.)

3.6 Teemat

Kuten luvussa 2 kuvataan sivuston yhtenäisyyden merkitystä, ASP.NET 2.0 -versiolla on mahdollista luoda visuaalista yhtenäisyyttä sivustolle teemoilla. Teemat voivat koostua CSS-tiedostoista ja .skin-päätteisistä (kuoret) tiedostoista sekä kuvista. Teemat sijoitetaan App_Themes-kansioon esimerkiksi alla olevan kuvan osoittamalla tavalla (Kuvio 7).

Kuvassa App_Themes-kansioon on luotu kaksi teemaa –Kesa ja Talvi– molemmat teemat sisältävät teeman mukaan nimetyn CSS- ja .skin-tiedoston, lisäksi Kesa-teemakansio sisältää Images nimisen kansion, joka sisältää teeman käytettävissä olevat kuvat. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 307.)



Kuvio 7. Teemat

CSS-tiedostot toimivat samalla tavalla ASP.NETin yhteydessä kuin tavallisillakin HTML-sivuilla. CSS:llä voidaan vaikuttaa esimerkiksi HTML-elementtien asemointiin ja tekstin muotoiluun. Koska CSS:llä ei voida vaikuttaa suoraan ASP.NETissä käytettäviin palvelinpuolen kontrolleihin, on kontrolleja varten olemassa .skin-tiedostot. Esimerkissä on kuvattu kalenteri-kontrollin muotoilu skin-tiedostossa. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 315.)

Esimerkki:

```
<asp:Calendar runat="server" BackColor="#FFFFCC"
        BorderColor="#FFCC66" BorderWidth="1px"
        DayNameFormat="Shortest"
        Font-Names="Verdana" Font-Size="8pt" ForeColor="#663399"
        Height="300px" ShowGridLines="False" Width="300px">
        <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True"/>
        <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
        <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
        <DayHeaderStyle  BackColor="#FFCC66" Font-Bold="True"
        Height="1px" />
        <TitleStyle CssClass="teema_Yla" BackColor="#990000"
        Font-Bold="False" Font-Size="12pt" ForeColor="#FFFFCC" />
</asp:Calendar>
```

Kontrollien ja HTML-elementtien muotoilu voidaan tehdä myös jokaiselle sivulle erikseen, mutta silloin sivuston visuaalisuuden kehittämisen hallittavuus heikkenee hyvinkin paljon. Luomalla teemoja voidaan sivuston ilmettä muokata yhdessä paikassa olevilla tiedostoilla. Jos sivustolla on käytössä kirjautuminen, rekisteröityminen tai oma profiili, voidaan käyttäjille antaa mahdollisuus valita haluamansa teema käytettävissä olevista teemoista, jolloin käyttäjän kirjautuessa sivustolle, sivuston ulkoasu esitetään hänen valitsemansa teeman mukaan. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 307-322.)

3.7 Navigointi

ASP.NETin navigointijärjestelmä perustuu SiteMap-provideriin, joka mahdollistaa nopean ja helpon tavan luoda sivustolle navigointi. Providereista on kerrottu lisää luvussa 3.10. ASP.NET 2.0 tarjoaa kolme erilaista sivuston navigointiin liittyvää kontrollia: Menu, TreeView ja SiteMapPath (”Bread Crums/leivänmuru”). (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 640, 646 – 647, 667 – 668.)

Menu-valikko on navigointivalikko, jolla saadaan sivuston navigointi rakennettua pieneen tilaan (Kuvio 8). Ideana menu-valikossa on, että käyttäjän viedessä hiiren menun päälle aukeaa näkyviin kyseiseen aiheeseen kuuluvat sivulinkit näkyviin. TreeView-navigointi kontrollilla kuvataan sivuston rakenne puumaisena. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 646 –647, 667 – 668.)

Kuvio 8. Menu-ja TreeView -valikot

Web.sitemap-tiedosto on XML-pohjainen tiedosto, johon kuvataan sivuston tiedostorakenne. Tiedosto koostuu <siteMap> -elementistä ja <siteMapNode>-elementeistä. Juurielementti <siteMap> voi olla vain yhden kerran tiedostossaan. Elementti <siteMapNode> koostuu kolmesta attribuutista title (otsikko), description (kuvaus) ja url (tiedostopolku). Title-attribuutin arvolla annetaan kuvaus linkistä. Description-attribuutin arvolla voidaan antaa kuvaus linkistä ja käyttää sitä myös ToolTip-ominaisuutena. Url-attribuutin arvoksi annetaan avattavan sivun osoite. Alla olevassa esimerkissä on kuvattu sitemap-tiedosto. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 638.)

Esimerkki:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="~/Etusivu.aspx" title="Etusivu"  description="Etusivu">
      <siteMapNode url="~/Ajankohtaista.aspx" title="Ajankohtaista"  description="" >
    <siteMapNode url="~/Tapahtumat.aspx" title="Tapahtumat"
          description="Tietoa tulevista tapahtumista" />
    <siteMapNode url="~/Tiedotteet.aspx" title="Tiedotteet"
          description="Tärkeitä tiedotteita" />
    <siteMapNode url="~/Muuta.aspx" title="Muuta"
          description="Muita tärkeitä asioita" />
    </siteMapNode>
    <siteMapNode url="~/Omat.aspx" title="Omat tiedot"  description="" >
    <siteMapNode url="~/SalasananVaihto.aspx" title="Salasanan vaihto"
          description="Sivu salasanan vaihtoon" />
    <siteMapNode url="~/OmaProfiili.aspx" title="Omat tiedot"
          description="Oman profiilin tiedot" />
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

Ilman että Menu ja TreeView kontrolleihin kirjoitettaisiin suoraan sivustorakenne kontrollien ominaisuuksiin, voidaan ne yhdistää sitemap-tiedostoon (sivukartta) SiteMapDataSource (tietosidonnasta ja siihen liittyvistä kontrolleista lisää luvussa 3.9) komponenteilla. Kolmas navigointi kontrolli, SiteMapPath, osaa yhdistää itsensä suoraan sivukartta tiedostoon. SiteMapPath-kontrollilla näytetään käyttäjän sijainti sivuston hierarkkisessa

rakenteessa. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 640, 646 – 647, 668.)

3.8 Palvelinkontrollit

ASP.NETissä käytössä olevat palvelinkontrollit voidaan jakaa karkeasti kahteen eri ryhmään: HTML-kontrolleihin ja Web-palvelinkontrolleihin. Palvelintoiminnallisuuksilla varustetut HTML-kontrollit ovat normaaleja HTML-merkkauskielen elementtejä, joita voidaan käsitellä ohjelmallisesti ASP.NET sovelluksessa. HTML-elementit saadaan palvelinpohjaiseksi lisäämällä HTML-kontrollille runat=”server”-attribuutti. Alla olevassa esimerkissä on sivulle sijoitettu teksti-kenttä ja nappi. Napille on määritelty Button1_ServerClick-tapahtuma, jossa käyttäjän painaessa nappia ”vastataan” viestillä Terve ja hänen tekstikenttään kirjoittamallaan nimellään. (Inkinen, V. 2003, 134.)

Esimerkki:

```
<script runat="server">
    protected void Button1_ServerClick(object sender, EventArgs e)
    {
        Response.Write("Terve " + Text1.Value);
    }
</script>

  <form id="form1" runat="server">
    <div>
    Anna nimesi:
        <input id="Text1"  type="text" runat=server />
        <input id="Button1" type="button"
        value="Lähetä" runat=server
        onserverclick="Button1_ServerClick" />
    </div>
  </form>
```

Toinen ryhmä eli Web-palvelinkontrollit, muodostuvat ASP.NETissä käytössä olevista kontrolleista. Kontrollit ovat toiminnallisuudeltaan rikkaampia kuin HTML-kontrollit. Alla olevassa esimerkissä on samanlainen sivu kuin on HTML-esimerkissä, mutta kontrolleina on käytetty ASP.NETin omia kontrolleja. Erona koodissa on kontrollien merkkaus, ASP.NETissä käytetään prefiksinä asp:tä (<asp:TextBox…). (Inkinen, V. 2003, 144.)

Esimerkki:

```
<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Write("Terve " + TextBox1.Text);
    }
</script>
 <form id="form1" runat="server">
    <div>
    Anna nimesi: <br />
      <asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
      <asp:Button ID="Button1" runat="server"
      Text="Lähetä" OnClick="Button1_Click" />
       </div>
 </form>
```

Palvelinkontrollit sisältävät monia samantapaisia kontrolleja kuin on HTML-kontrolleissakin esimerkiksi tekstikenttä-, nappi-, valinta- ja kuvakontrollit. Näiden lisäksi ASP.NET sisältää myös ”täysin” valmiita kontrolleja kuten kalenteri, AdRotator ja ”velho”-kontrollin jonka avulla voidaan ohjata käyttäjää toimimaan tietyn kaavan mukaan. ASP.NETissä on myös käytössä erilaisia käyttäjän syötteiden tarkistamiseen tarkoitettuja komponentteja. Tarkistus-komponentit (Validation) on kuvattu liitteessä (Liite 3). (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 155 –156, 190, 199, 229 – 230, 240.)

Lisäksi ASP.NET sisältää myös erityisesti tietovarastoissa olevien tietojen näyttämiseen ja muokkaamiseen tarkoitettuja Data-kontrolleja. Esimerkiksi GridView-kontrolli on tarkoitettu erityisesti suuren tietomäärän näyttämiseen kerralla. Kontrolli sisältää valmiiksi rakennetut tietojen lajittelun- ja sivutustoiminnot sekä tietueen valinnan. Data-kontrollit ”työskentelevät” tietolähdekomponenttien kanssa, joista on kerrottu lisää seuraavassa luvussa. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 389 – 390.)

Myös omien kontrollien tekeminen on mahdollista. Kontrollit voivat sisältää ASP.NETin omia kontrolleja ja niiden tapahtumia sekä toimintoja. Tehtyjä kontrolleja voidaan käyttää monta kertaa web-sovelluksessa, jolloin sivuston kehittäminen helpottuu huomattavasti. Kontrolli liitetään sivulle @Register-direktiivillä, ja lomakkeella käytetään prefiksinä määriteltyä nimeä. Alla oleva esimerkki kuvaa luodun kontrollin käyttöä sivulla. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 1023, 1037.)

Esimerkki:

```
<%@ Register Src="OmaKontrolli.ascx" TagName="OmaKontrolli"
TagPrefix="uc2" %>
<form id="form1" runat="server">
        <div>
        <uc2:OmaKontrolli ID="OmaKontrolli1" runat="server" />
        </div>
</form>
```

3.9 Tietolähteet

ASP.NET sisältää tietolähdekomponentteja (DataSource), joiden avulla sivustolla käytettävä tieto voidaan helposti työstää erilaisten tietolähteiden kanssa, esimerkiksi tietokannan, XML tiedostojen, tai jopa sovelluskerroksen kanssa. Tietolähdekomponenttien avulla tietojen hakeminen ja niiden sidonta erilaisille kontrolleille on erittäin helppoa ilman koodausta, varsinkin jos käytettävissä on jokin Visual Studio 2005 versio, jonka avulla kontrollit ja komponentit voidaan luoda ”vedä ja pudota”-menetelmällä lomak-

keelle, jolloin ohjattu toiminto avustaa tietolähteen yhdistämisessä tietovarastoon. tolähdekomponenttien avulla tietoa voidaan myös muokata. Alla olevassa kuviossa (Kuvio 9) on listattu tietolähdekomponentit sekä niiden käyttökohteet (MSDN c.)

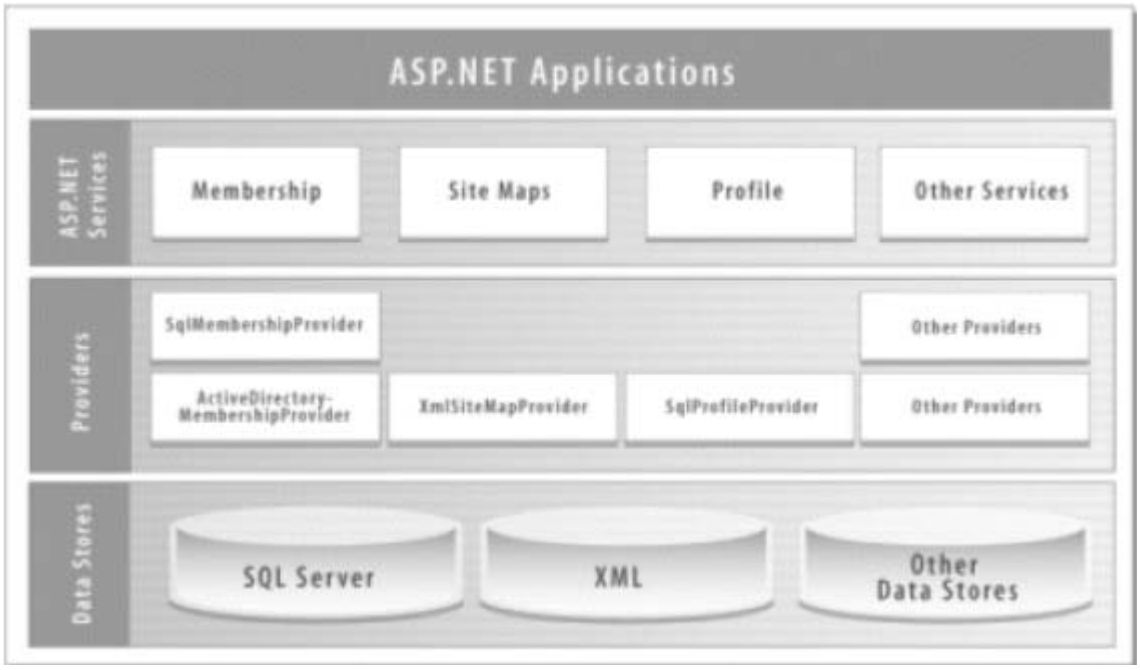
ObjectDataSource	Kontrolli työskentelee sovelluskerroksen kanssa. Kontrolli on suunniteltu toimimaan objektien kanssa jotka toteuttavat yhden tai useamman metodin tiedon hakemiseen tai sen muokkaamiseen.
SqlDataSource	Kontrolli hakee ja muokkaa tietoa käyttäen SQL komentoja. Kontrolli toimii Microsoft SQL Server, OLE DB, ODBC, ja Oracle tietokantojen kanssa.
AccessDataSource	Kontrolli on erikoisversio SqlDataSource kontrollista. AccessDataSource kontrolli on tarkoitettu työskentelyyn Access-tietokantojen kanssa.
XmlDataSource	Kontrolli lukee ja kirjoittaa XML muotoista tietoa.
SiteMapDataSource	Kontrolli työskentelee sivukarttojen (sitemaps) kanssa ja tarjoaa sivun navigointiin tarvittavat tiedot. Yleisimmät käyttökohteet ovatkin juuri navigointiin tarkoitetut Menu- ja TreeView-kontrollit.

Kuvio 9. Tietolähdekomponentit (MSDN c.)

Suuri hyöty tietolähde kontrolleista saadaan kun ne yhdistetään johonkin ASP.NET palvelinkontrolliin. Yksi ASP.NET 2.0 version myötä tullut kontrolli on GridView datakontrolli, joka on tarkoitettu suuren tietomäärän näyttämiseen ja muokkaamiseen. GridView-kontrolli sisältää valmiiksi erilaisia toimintoja kuten, tietojen lajittelun, sivutuksen, poistamisen, muokkaamisen jne. edellyttäen tietysti, että esimerkiksi poistamiselle ja muokkaamiselle on tehty tarvittavat metodit tai SQL-komennot tietolähteelle. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 389 – 390.)

3.10 Provider-malli

ASP.NET 2.0 sisältää tilanhallintaan palveluja, joiden avulla tiedot voidaan tallentaa tietokantaan tai muihin eri medioihin. Nämä palvelut pohjautuvat Provider-malliin (kuvio 10). Provider on malli, joka tarjoaa yhtenäisen rajapinnan palveluiden ja tallennettujen tietojen välille. (MSDN 2005 a.)



Kuvio 10. Provider-malli (MSDN 2005 a.)

Provider-malli sisältää kahdeksan ASP.NET palvelua joissa jokaisella on yksi tai useampi provider. Kuviossa 11 on kuvattu palvelut ja niitä käyttävien providerien nimiavaruuden. Provider-malli sisältää valmiita palveluja joita voi hyödyntää sivustoa tehdessä, nämä palvelut käyttävät providereita (esim. SqlMembershipProvider, SqlRoleProvider) oletuksena SQL Server Express Edition - tietokantaa tiedon tallentamiseen. Kun taas InProcSessionStateStore-Provider tallentaa tiedot palvelimen muistiin ja XmlSiteMapProvider käyttää XML-tiedostoja tiedon tallentamiseen. (MSDN 2005 a.)

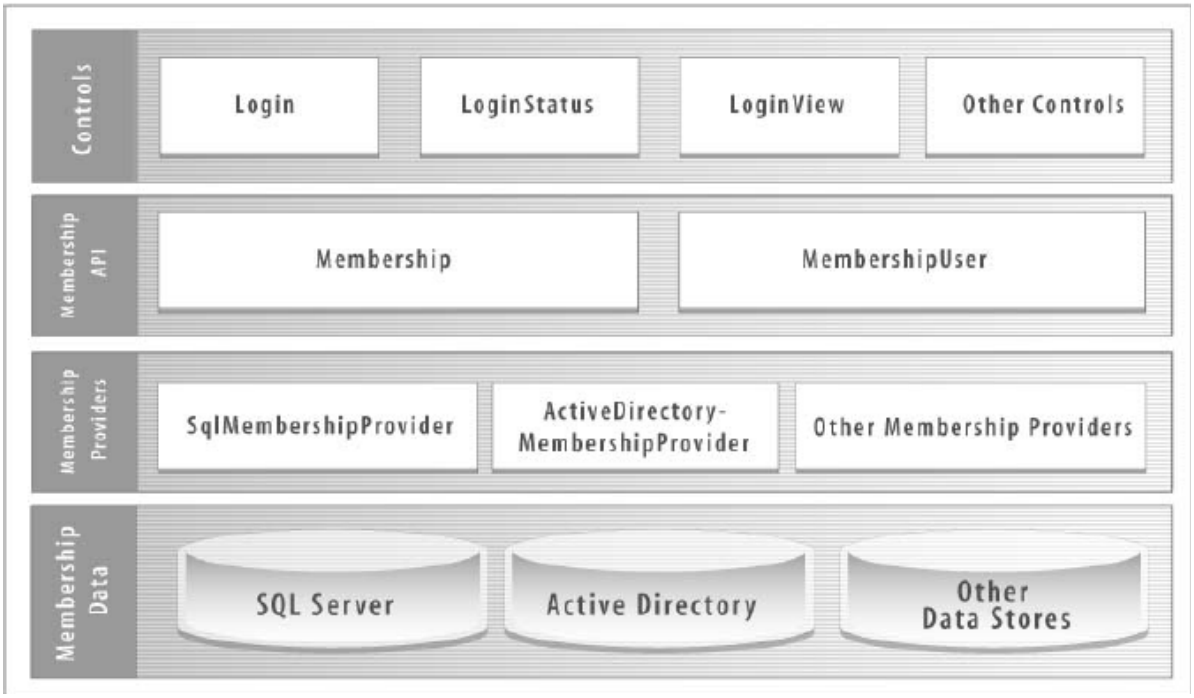
Membership	System.Web.Security.ActiveDirectoryMembershipProvider System.Web.Security.ActiveDirectoryMembershipProvider
Role management	System.Web.Security.AuthorizationStoreRoleProvider System.Web.Security.SqlRoleProvider System.Web.Security.WindowsTokenRoleProvider
Site map	System.Web.XmlSiteMapProvider
Profile	System.Web.Profile.SqlProfileProvider
Session state	System.Web.SessionState.InProcSessionStateStore System.Web.SessionState.OutOfProcSessionStateStore System.Web.SessionState.SqlSessionStateStore
Web events	System.Web.Management.EventLogWebEventProvider System.Web.Management.SimpleMailWebEventProvider System.Web.Management.TemplatedMailWebEventProvider System.Web.Management.SqlWebEventProvider System.Web.Management.TraceWebEventProvider System.Web.Management.WmiWebEventProvider
Web Parts personalization	System.Web.UI.WebControls.WebParts.SqlPersonalizationProvider
Protected configuration	System.Configuration.DPAPIProtectedConfigurationProvider System.Configuration.RSAProtectedConfigurationProvider

Kuvio 11. Provider-mallin palvelut (MSDN 2005 a.)

Koska jokainen provider on johdettu ProviderBase-nimisestä luokasta, on myös omien providerien tekeminen mahdollista. Tällä saadaan etuna se, että tiedot voidaan tallentaa myös muihin tietokantoihin kuin vain Microsoftin omiin SQLServer-versioihin, kuten Oraclen tai MySql tietokantoihin. (MSDN 2005 a.)

Osaan palveluista provider-malli tarjoaa myös valmiita käyttöliittymä-kontrolleja. Kuviossa 12 on kuvattu Membership-palvelu, jossa ylimmällä kerroksella on käyttöliittymä-kontrollit Login, LoginStatus, LoginView ja muut kontrollit joihin sisältyvät salasanan vaihto, salasana palautus ja käyttäjän luomiseen tarkoitetut ohjatut toiminnot. Kontrollien alapuolella oleva kerros tarjoaa kontrollien käyttämän API:n Membership-palveluun.

Membership-palvelu tallentaa käyttäjän tiedot johonkin tietolähteeseen käytettävän providerin välityksellä. (MSDN 2005 a.)



Kuvio 12. Membership-palvelu (MSDN 2005 a.)

3.11 Tietoturva

Kun Internetiin lisätään sivusto, ovat sivut kaikkien sivustolle tulevien avattavissa ja katseltavissa. Joskus on kuitenkin, varsinkin tehtäessä sivustoja, joiden avulla myös lisätään ja muokataan näytettävää tietoa, suojattava osa sivustolla olevista tiedoista ja sivuista, mikäli ne sisältävät luottamuksellista tietoa ja näin ollen ei saa olla kaikkien saatavilla. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 835.)

ASP.NETissä on kolme erilaista tunnistautumismenetelmää (Authentication), Windows, Forms ja Passport. Web-sovelluksen web.config tiedostossa määritellään sovelluksen käyttämä tunnistautumismenetelmä. Tunnistatumismenetelmillä tarkistetaan onko käyttäjä juuri se henkilö joka hän väittää olevansa. Alla olevassa esimerkissä on kuvattu tunnistautumismenetelmän käyttöönotto web.config-tiedostossa.

Esimerkki:

```
<system.web>
        <authentication mode="Passport"|"Windows"|"Forms"/>
</system.web>
```

(Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 837 – 838.)

Windows-tunnistautuminen käyttää nimensä mukaisesti Windows ympäristöön luotujen toimialueiden käyttäjäryhmiä ja niihin luotuja käyttäjätilejä. Windows-pohjainen tunnistautuminen soveltuukin parhaiten esimerkiksi yrityksen intranet-ympäristöön. Windows-tunnistautuminen suoritetaan IIS:n kanssa, joka käyttää tarkistustyyppin menetelmään jotakin kolmesta tavata: Basic, Digest tai Integrated Windows. Näistä menetelmistä tarkoituksen mukaisin määritellään IIS asetuksissa. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 838 – 839, 843 – 845.)

Forms-pohjaisessa tunnistautumismenetelmässä tunnistus tapahtuu lomakkeella. Tämän menetelmän avulla sivustolle on mahdollista rakentaa oma kirjautumis- ja käyttäjienhallintamenetelmä Membership-, Role- ja Profile-palveluita ja niiden providereita hyväksikäyttäen. Käyttäjien tiedot on mahdollista tallentaa esimerkiksi tietokantaan. ASP.NET 2.0n mukana tulevilla käyttöliittymä-kontrolleilla (Login, LoginStatus, LoginView, salasanan vaihto, salasana palautus ja käyttäjän luomiseen tarkoitetut ohjatut toiminnot) mahdollistetaan käyttäjätietojen hallinta varsin helpoksi luoda. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 732, 845, 853.)

Passport-tunnistaumismenetelmässä sivustolla voidaan käyttää Microsoftin Passport-palvelua, jonka avulla käyttäjä voi yhdellä kirjautumisella käyttää kaikkia Passport-tunnistautumista käyttäviä sivustoja ja sovelluksia. Kun sivusto on määritelty käyttämään Passport-tunnistautumista käyttäjä ohjataan Microsoftin Passport-sivustolle kirjautumista varten. Jos tunnistautuminen onnistuu, käyttäjä ohjataan takaisin alkuperäiselle sivustolle. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 855.)

Kun käyttäjä on tunnistettu, käyttöoikeuksilla (Authorization) määritellään, mihin tietoihin tai sivuihin hänellä on oikeus päästä. Kuten yleensä esimerkiksi Windows ympäristössä luodut käyttäjät liitetään johonkin rooliin ja roolille annetaan oikeuksia tehdä tiettyjä toimia kyseisessä ympäristössä. Samaa periaatetta voidaan soveltaa myös

web-sivustolla. Rooleille annetaan oikeudet nähdä vain kyseiselle roolille tarkoitetut sivut ja helpoin tapa on määritellä tämä web-config tiedostoon. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 732, 773.)

Alla oleva esimerkki on web-config tiedostosta johon on merkitty sivun nimi ”Omat.aspx” ja sille määritelty käyttöoikeus vain käyttäjäryhmälle ”Jasen”. Eli käyttäjillä, jotka kuuluvat käyttäjäryhmään Jasen, on kirjautumisen jälkeen oikeus päästä kyseiselle sivulle.

Esimerkki:

```
<location path="Omat.aspx">
  <system.web>
    <authorization>
      <allow roles="Jasen"/>
      <deny users="*"/>
    </authorization>
  </system.web>
</location>
```

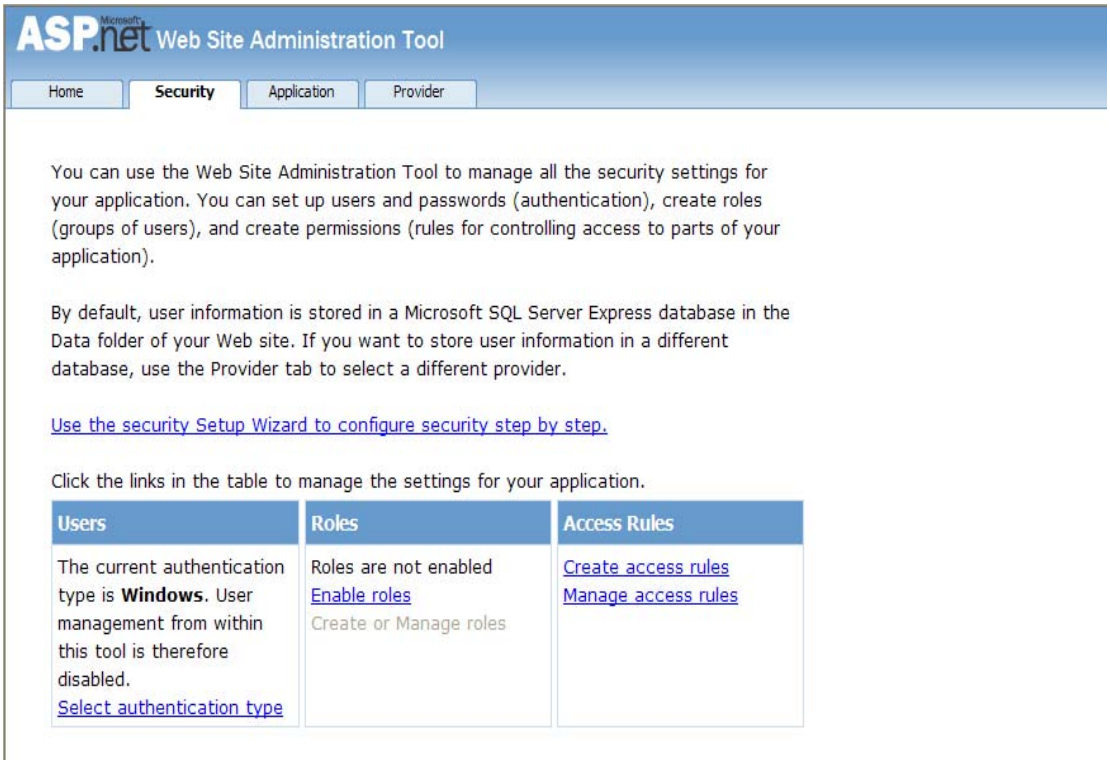
Toki on järkevämpää luoda kehitettäessä sivustoa oma kansio esimerkiksi jäsenille, jonne sijoitetaan sivut joihin kyseisellä ryhmällä on oikeus päästä. Tällöin jokaiseen kansioon luodaan uusi web.config tiedosto, johon määritellään käyttöoikeudet.

Luvussa 3.7 esiteltyjen navigointiin liittyviä ominaisuuksia on mahdollista käyttää hyväksi myös sivuston turvallisuudessa. Tällöin mahdollistetaan, että halutut sivut näkyvät navigointi valikoissa vain niille käyttäjille, joilla on oikeus nähdä ne. Alla olevassa web.config esimerkissä on siteMap-providerin securityTimmingEnabled arvoksi annettu true, jolloin ASP.NET huolehtii annettujen käyttöoikeuksien mukaisesti navigoinnissa näkyvät linkit. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 688.)

Esimerkki:

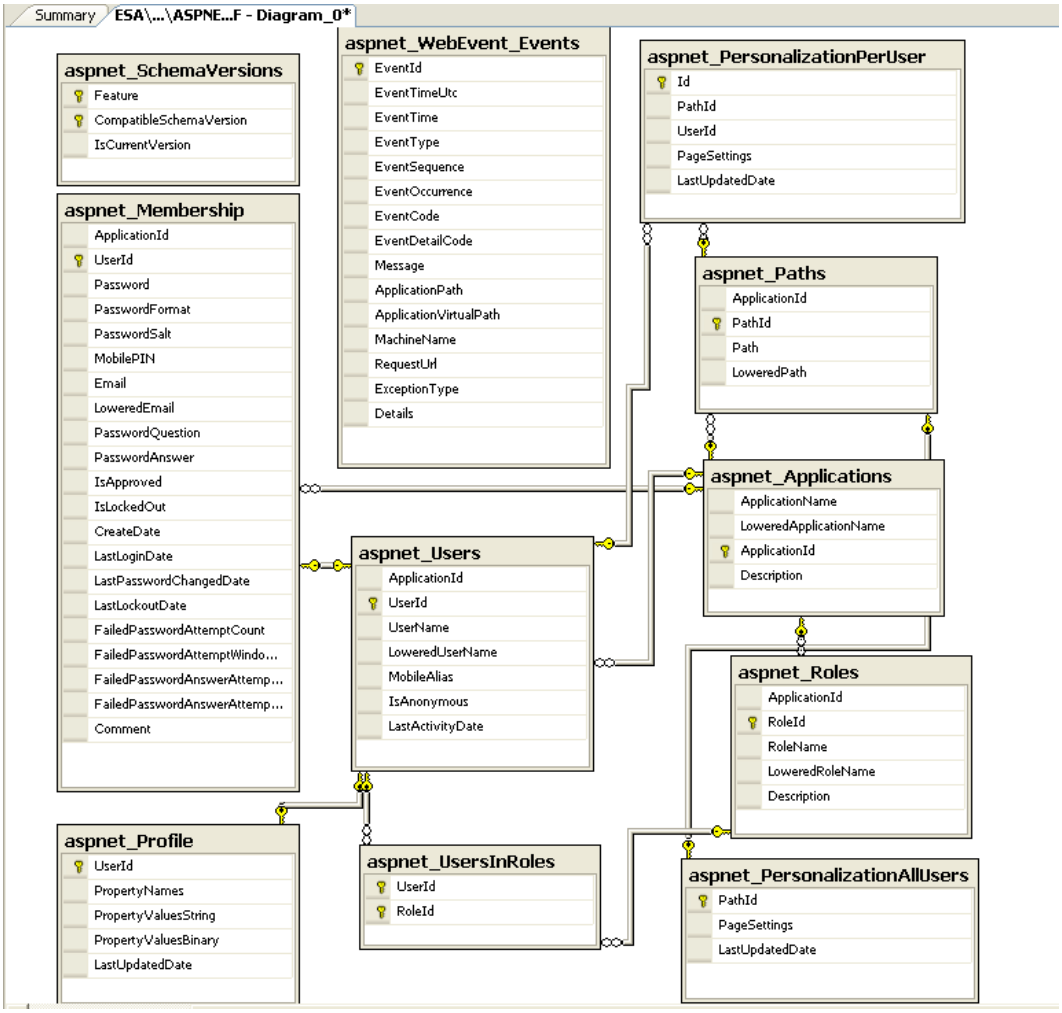
```
<siteMap defaultProvider="XmlSiteMapProvider" enabled="true">
    <providers>
      <add  name="XMLSiteMapProvider"
          description="Oletus navigointi-provider"
          siteMapFile="Web.sitemap"
          securityTrimmingEnabled="true"
          type="System.Web.XmlSiteMapProvider"/>
    </providers>
</siteMap>
```

Visual Web Developer 2005 Express Editionin mukana tulee työkalu (Kuvio 13.), jonka avulla voidaan luoda ja hallita sivustolle tulevia tietoturvamäärityksiä. Sovelluksen avulla sivustolle voidaan määritellä käyttäjäryhmiä, luoda käyttäjätilejä ja antaa käyttöoikeuksia haluttuihin kansioihin tai sivuihin. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 731, 835; Khosravi, S. 1342 – 1343.)



Kuvio 13. Web Site Administration Tool-työkalu

Web Site Administration Tool-työkalua käytettäessä, se luo oletuksena Microsoft SQL Server Express tietokannan (Kuvio 14.) App_Data-kansioon ja tallentaa luotuun tietokantaa käyttäjätiedot. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 731, 835; Khosravi, S. 1342 – 1343.)



Kuvio 14. Käyttäjätietokannan taulut ja yhteydet

3.12 Caching

Dynaamisissa sivustoissa, kuten myös tavallisissa sivustoissa on tärkeää myös sivuston sivujen nopea latautuminen. ASP.NETissä sivujen latautumiseen on mahdollista vaikuttaa väliaikaismuistikäsittelyllä (Caching). Kun sivulla hyödynnetään cachingiä, on sivun

esittäminen nopeampaa muistista kuin, että sivu luotaisiin uudelleen joka kerta, kun pyyntö tulee palvelimelle. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 907.)

ASP.NETillä on mahdollista sijoittaa palvelimen muistiin joko koko sivu tai jokin tietty osa sivusta. Tällöin esimerkiksi, kun sivulla käytetään montaa itse tehtyä kontrollia, voidaan hyvin tarkkaan hallita sitä tietoa mikä latautuu muistista ja mikä tieto haetaan esimerkiksi tietokannasta käyttäjän pyytäessä sivua palvelimelta. (Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006, 907 – 908, 910 – 911.)

4 KAINUUN KISSANYSTÄVÄT RY:N INTERNET-SIVUSTON TOTEUTUS

Kainuun Kissanystävät ry on vuonna 1991 perustettu ja vuonna 2006 rekisteröitynyt yleishyödyllinen yhdistys, jonka tarkoituksena on hoitaa ja uudelleen sijoittaa löytyneitä, kodittomia kissoja. Yhdistys huolehtii kissojen terveydestä ja leikkuuttaa aikuiset kissan ennen uudelleen luovuttamista. Yhdistyksen perusajatuksena on, ettei yhtäkään kissaa tarvitse lopettaa tarpeettomasti, mikäli kissa voidaan hoitaa terveytensä puolesta kuntoon. Yhdistys toimii vapaaehtoistyön varassa ja rahoittaa toimintansa lähinnä jäsenmaksuilla ja lahjoituksilla. (Rautiainen 30.6.2007.)

Yhdistyksen toiminta tukeutuu tiedottamiseen ja tiedon välittämiseen ”kissanystävien” kesken. Toiminnan keskeisenä välineenä on nopea tiedon kulku yhdistyksen ja kissoja mahdollisesti itselleen haluavien välillä, sillä sijaiskoteja on usein vähemmän kuin apua tarvitsevia kissoja. Tietoa pyritään välittämään usein jo siinä vaiheessa, kun kissa on otettu hoitoon Löytöeläintarha Juliukseen, jossa kissaa on pidettävä 15 vuorokautta. Eläintarhalla ei ole velvollisuutta jatkaa kissan hoitamista 15 vuorokauden jälkeen, jolloin kissa voidaan lopettaa, mikäli sille ei löydy uutta kotia. Nopea tiedon kulku on siten tärkeää heti, kun tieto löydetystä kissasta saadaan jollekin yhdistyksen jäsenelle. (Rautiainen 30.6.2007.)

Olennaisin osa opinnäytetyötä on Kainuun Kissanystävät ry:n Internet-sivuston muuttaminen osittain dynaamiseksi ASP.NET tekniikkaa ja SQL Server -tietokantaa hyväksi käyttäen. Yhdistyksen hallituksen jäsenet voivat tallentaa tietoa löydetyistä ja hoitoon otetuista kissoista sekä yhdistyksen jäsenistä tietokantaan. Tietokannan kautta tarvittavat tiedot kissoista päivittyvät kaikkien sivustoa käyttävien henkilöiden nähtäville.

4.1 Nykytilanne

Nykyisin Kainuun Kissanystävät ry:llä ei ole varsinaisia rekistereitä löydetyistä ja hoitoon otetuista kissoista. Jäsenrekisteri on tällä hetkellä Excel-taulukossa. Internet-sivut ovat staattiset ja yhden henkilön päivitettävissä ja vastuulla, jolloin on mahdollista, että esimerkiksi tietoliikenneyhteyden katkoksen takia sivustoa ei voi päivittää. Lisäksi vaarana on, että vastuuhenkilön ollessa estynyt hoitamaan tehtäväänsä tiedon välittäminen katkeaa. Opinnäytetyön tarkoituksena on muuttaa sivuston hallintaa siten, että useammalla henkilöllä on oikeus päivittää tarvittavat tiedot kissoista sivustolle, näin ollen vastuuta kyseisestä työstä saadaan jaettua. (Rautiainen 30.6.2007.)

Kainuun Kissanystävien nykyiset Internet-sivustot koostuvat 15:stä eri sivusta, joista opinnäytetyön kannalta tärkeimmät ovat sivut, joissa kerrotaan kadonneista, löytyneistä, kotia etsivistä, sekä kodin löytäneistä kissoista. Kehittämistyössä kyseiset sivut muutettiin tietokantapohjaisiksi, jolloin hallituksen jäsenten tietokantaan laittamat tiedot kissoista näkyvät kyseisillä sivuilla.

Tietoja kissoista on kerätty lähinnä luovutushetkellä tehtävistä luovutussopimuksista ja esimerkiksi hoitopäivät kukin sijaiskoti on kerännyt itsenäisesti, joten tietoa on säilytetty hajanaisesti. Kerätty tieto on kuitenkin oleellista koota yhteen yhdistyksen toiminnan arvioimista ja tulevan toiminnan suunnittelua varten. (Rautiainen 30.6.2007.)

Yhdistyksessä jäseninä olevien henkilöiden tietoja on kerätty Excel-taulukossa pidettyyn rekisteriin. Ongelmana on ollut, että esimerkiksi sihteerin vaihtuessa edellisten vuosien jäsenrekisterit eivät välttämättä tule uuden sihteerin käyttöön ongelmitta. Lisäksi jäsenet eivät ilmoita yhteystietojaan vuosittain maksaessaan jäsenmaksua, joten ongelmana on ollut yhteystietojen kerääminen edellisvuosien taulukoista, yhtenäisellä jäsenrekisterillä tämäkin ongelma poistuu. (Rautiainen 30.6.2007.)

4.2 Suunnittelu ja toteutus

Suunnitteluvaiheessa kartoitettiin nykytilanne ja tarvittavat muutoskohteet, lähtökohtana olivat jo käytössä olevat sivut ja niillä valmiiksi oleva tieto. Yhteistyössä yhdistyksen

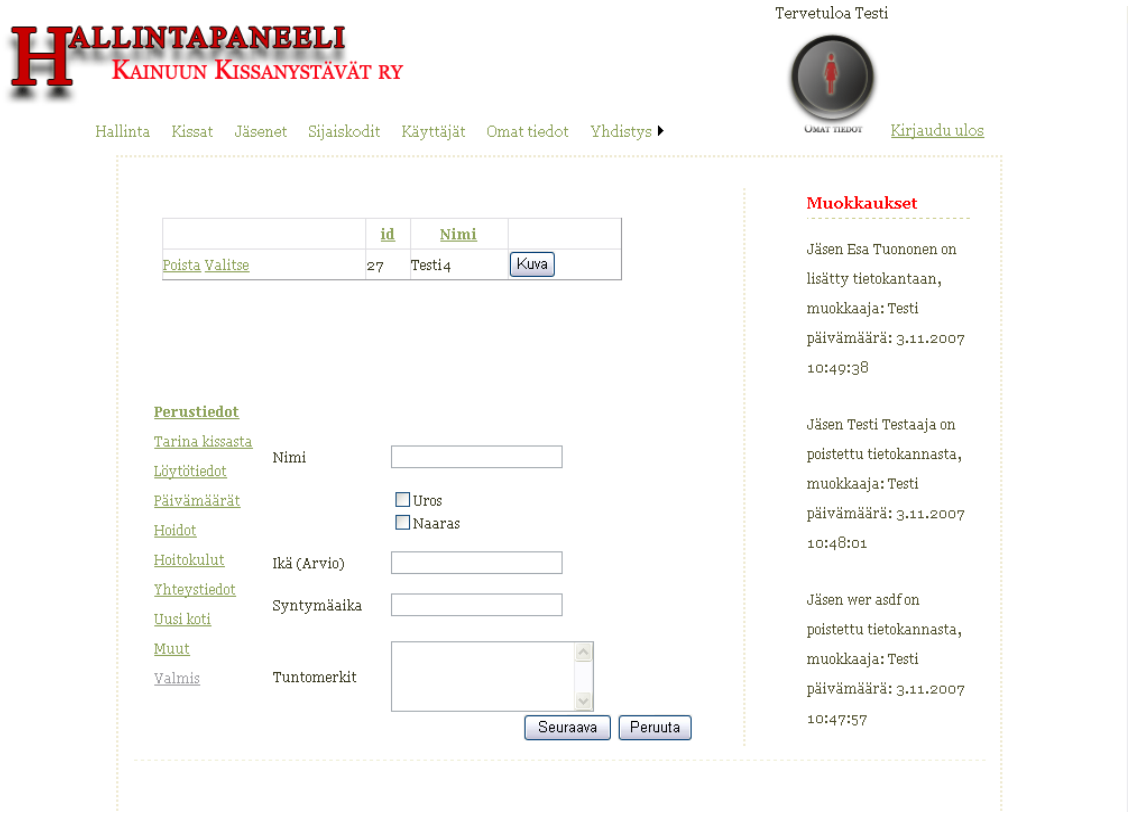
hallituksen jäsenten kanssa laadittiin suunnitelma, jossa käytiin läpi tietokantaan tavat tiedot, niiden tarpeellisuus Internet-sivuston näkyvällä osiolla sekä ulkoasuun liittyviä ominaisuuksia. Resurssien vähäisyyden vuoksi päätettiin Internet-sivusto ja sivustolla tarvittava tietokanta tehdä ilmaisilla Microsoft Visual Web Developer 2005 Express Edition sekä Microsoft SQL Server 2005 Express Edition kehitysvälineitä hyödyntäen.

Sivusto on pyritty tekemään kerrosarkkitehtuurin mukaisesti. Tavoitteena oli saada jaettua kerrokset selkeisiin kokonaisuuksiin, jolloin myöhemmin tapahtuvat muutokset ja lisäykset olisivat mahdollisimman helposti tehtävissä. Sovellus on jaettu kolmeen eri kerrokseen: käyttöliittymä-, sovellus- ja tietokantakerros.

Käyttöliittymäkerros sisältää sivuston käyttäjälle näkyvimmän osan, eli palvelimella muodostettavan selaimessa esitettävän sivun. Sovelluskerroksen tarkoituksena on toimia rajapintana käyttöliittymäkerroksen ja tietokantakerroksen välillä. Sovelluskerroksella olevia toimintoja voidaan hyödyntää monelta eri sivulta ilman, että ne pitäisi tehdä jokaiselle sivulle erikseen. Kolmannella kerroksella on tietokanta ja tietokantaa hyödyntävät tallennetut proseduurit.

Sivustolla hyödynnettävä tietokanta sisältää käyttäjätietojen tallentamiseen tarvittavat taulut sekä jäsen- ja kissarekistereiden taulut. Jäsen- ja kissarekistereiden osalta tietokanta ei ole rakenteeltaan monimutkainen, vaan tarkoituksena oli hallittavuuden takia pitää tietokanta mahdollisimman yksinkertaisena. Lisäksi tietokantaa tehtäessä luotiin tallennettuja proseduureja, joiden avulla sovelluksen koodissa ei tarvitse kyselyjä luoda ohjelmallisesti, vaan sovelluksessa voidaan hyödyntää jo valmiiksi tehtyjä kyselyjä. Tallennettujen proseduurien avulla myös kyselyjen hallinta saatiin ”yhteen paikkaan”.

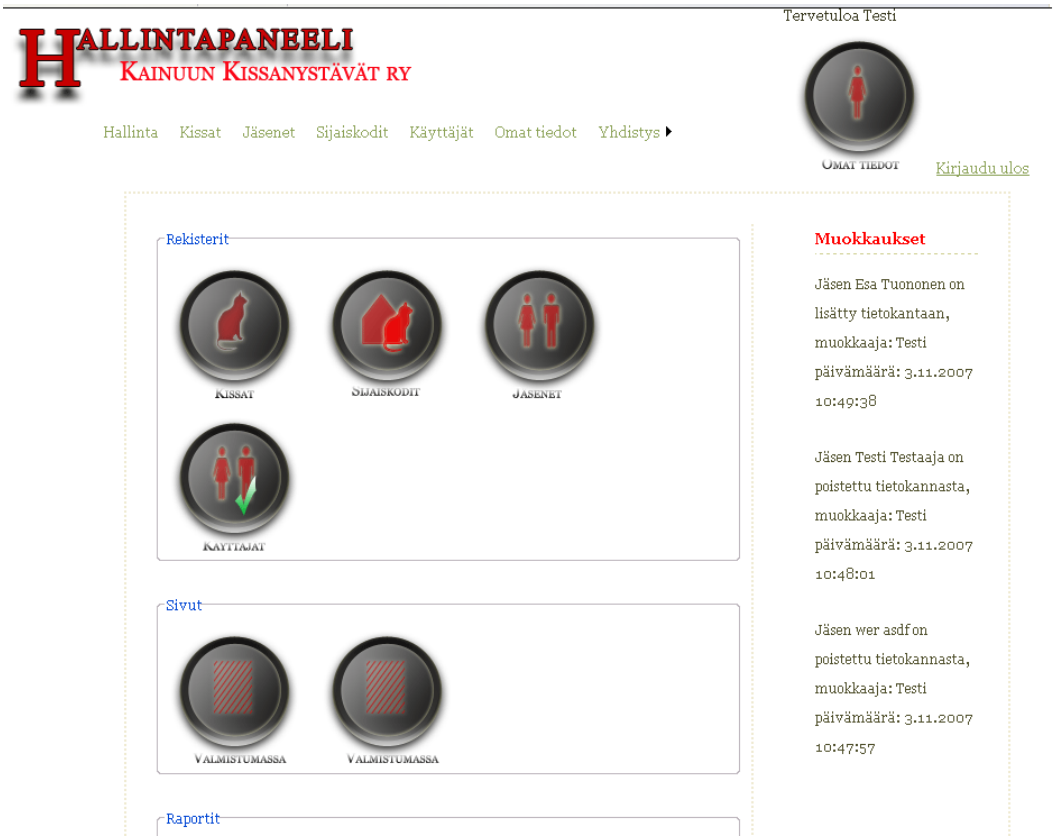
Tietokantaan tarvittaviksi tiedoiksi valittiin rekistereissä yleisestikin käytettäviä tietoja (nimi, osoite, sukupuoli yms.). Kerättävät tiedot rajattiin koskemaan perustietoja jäsenistä ja kissoista päätettiin kerätä laajempi tietokanta. Tietoja päätettiin kerätä kissarekisteriin toiminnan ja talouden suunnittelun kannalta olennaisia tietoja, esimerkiksi hoitopäivien laskentaa varten tarvittiin kissan tulo- ja lähtöpäivä. (Kuvio 15.)



Kuvio 15. Kainuun Kissanystävät ry:n kissarekisterin ylläpitosivu.

Tietokannan toteuttamisen jälkeen luotiin tarvittava hallintasivusto rekistereiden ylläpitoa varten. Hallintasivusto laadittiin mahdollisimman selkeäksi ja yksinkertaiseksi käyttää, jotta tietojen täyttäminen ja muokkaaminen onnistuu nopeasti ja helposti myös henkilöiltä, joilla ei ole paljon kokemusta Internet-ympäristöstä.

Hallintasivuston käyttöä on helpotettu muun muassa kuvapainikkeilla sekä jaottelemalla eri osioita omiksi erillisiksi asiakokonaisuuksiksi (Kuvio 16.). Lisäksi hallintasivustolla näytetään tehdyt, viimeisimmät muutokset, jotta vältytään päällekkäismuutoksilta usean henkilön käyttäessä hallintasivustoa jopa samanaikaisesti. Hallintasivuston pääsivu on tehty tulevaisuudessa mahdollisesti lisättäviä ominaisuuksia ajatellen, esimerkiksi pääsivulla on nähtävissä raporttiosio, johon voidaan myöhemmin lisätä uusia ”pikapainikkeita” erilaisia tulostettavia raportteja varten.

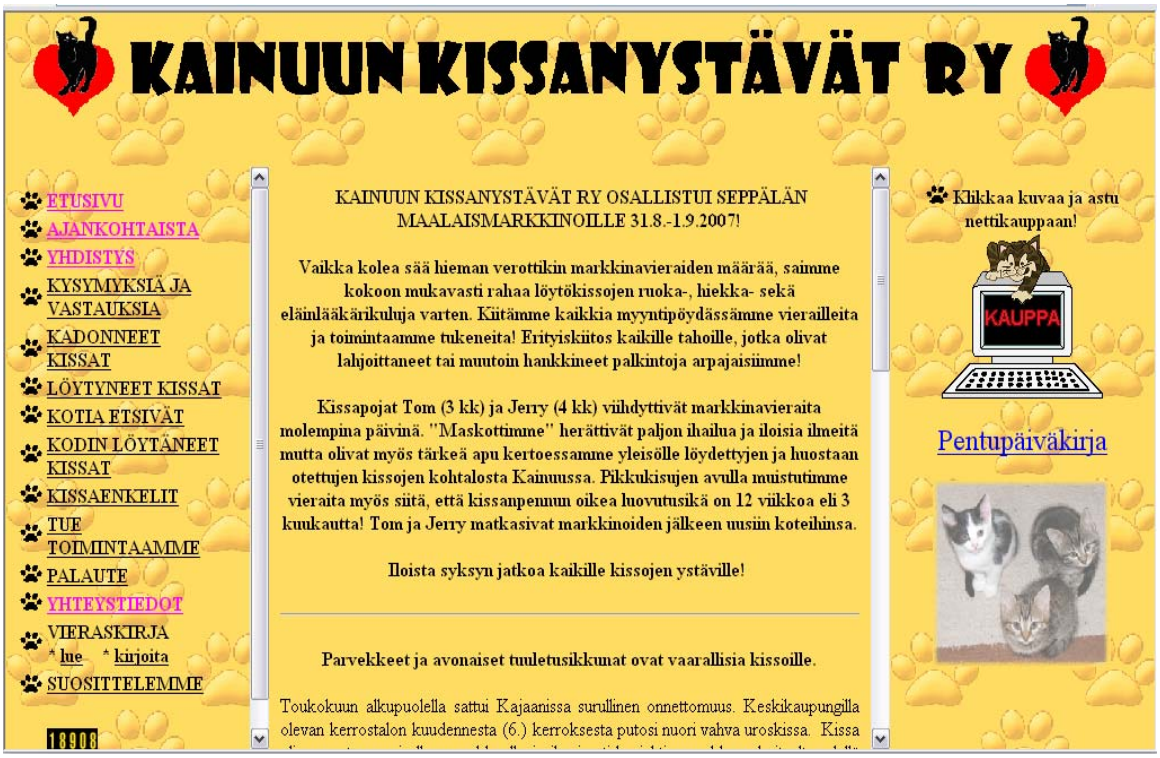


Kuvio 16. Kainuun Kissanystävät ry:n hallintasivuston pääsivu.

Sivustolle luotiin kirjautumis- ja käyttäjienhallintajärjestelmä, jonka avulla hallintasivustoille - tällä hetkellä rekisterit - on määritelty käyttäjäryhmät (tietyin käyttöoikeuksin). Käyttäjienhallintajärjestelmä tehtiin hyödyntäen luvussa 3.11 kerrottua sovellusta, joka loi käyttäjien tietojen (roolit, oikeudet yms.) tallentamiseen tarvittavan tietokannan taulineen automaattisesti. Sivustolle on määritelty käyttäjäryhmiä, joihin liitetään sivustoa otettaessa käyttöön omat käyttäjätilinsä, esimerkiksi hallituksen yksittäiselle jäsenelle luodaan oma käyttäjätili käyttäjäryhmään ”hallitus”.

Tietokannan toteuttamisen jälkeen luotiin tarvittavat sivu esimerkiksi kissojen tietojen esittämistä varten. Kun käyttäjä haluaa katsoa esimerkiksi kodin löytäneitä kissoja, sivuksi aukeaa pyydetyllä tiedolla kannasta löytyvät kissat, jos käyttäjä valitsee kotia etsivät kissat, näytetään tiedot tähän luokkaan merkityistä kissoista. Sivuja on siis olemassa vain yksi, jossa näytettävät tiedot vaihtuvat – aiemmin sivuja tarvittiin tähän tarkoitukseen viisi.

Nykyinen sivusto (Kuvio 17.) on tehty käyttämällä HTML-merkkausta, joka ei dollista tietokantojen hyödyntämistä. Koska uusi sivusto tehtiin ASP.NET -tekniikkaa hyödyntämällä, päätettiin muutoksen yhteydessä muuttaa sivuston ulkoasua muutenkin kuin rakenteelliselta osaltaan.



Kuvio 17. www.kissanystavat.net.

Sivuston yleistä ulkoasua muutettiin jättämällä pois taustakuvat ja muut kuin Kissa-osion kuvat. Lisäksi näkyvien linkkien määrää pienennettiin jakamalla sivustolla olevat sivut eri asiakokonaisuuksien alle, esimerkiksi linkkivalikon ”Kissat” alle siirrettiin osiot Kadonneet kissat, Löytyneet kissat, Kotia etsivät ja Kodin löytäneet kissat sekä Kissaenkelit (Kuvio 18.).

Vähentämällä kuvien ja näkyvien linkkien määrää pyrittiin luomaan selkeyden tuntua ja helpottamaan tiedon löytymistä sivustolla. Yhdistyksen käyttämä logo (kissa-sydän) liitetään myöhemmin yhdistyksen nimen viereen, logon avulla saadaan sivustolle lisää väriä ja huomionkiinnittäjä.

Kuvio 18. Kuva Kainuun Kissanystävät ry:lle kehitetystä Internet-sivustosta.

Yhdistyksen toiveiden mukaisesti sivustolle tehtiin oikeaan laitaan selkeästi erottuva oma osionsa tärkeimmille ”tärpeille”, osio näkyy sivustolla koko ajan. Osiolla mahdollistetaan nopea siirtyminen sivustolla mm. yhdistyksen toiminnan tukemisesta kiinnostuneet käyttäjät voivat siirtyä suoraan jäsenmaksuja ja lahjoituksia koskevalle sivulle. Lisäksi osioon haluttiin nähtäville tärkeitä yhteystietoja apua hakeville käyttäjille, muun muassa päivystävän eläinlääkärin puhelinnumero on nyt esillä, eikä sitä tarvitse erikseen etsiä eri linkkien takaa. Osiolla haluttiin herättää käyttäjien huomio myös muuhun yhdistyksen toimintaan kuin ”kissojen välittämiseen”.

5 POHDINTA

Yhdistysten rekistereiden – eritoten jäsenrekisterin – on oltava jo yhdistyslain puitteissa hyvin ylläpidettyjä ja ajan tasalla. Yhdistyksille olisi tärkeää laatia työkalu, jonka avulla tieto voidaan tallentaa ja käsitellä helposti sekä säilyttää ongelmitta. Yhdistyksille voi tulla eteen ongelmia mm. vastuuhenkilöiden vaihtuessa – jopa vuosittain, mikäli tietoa säilytetään vain henkilöiden omilla papereilla tai tiedostoissa.

Opinnäytetyötä tehtäessä työtä jouduttiin rajaamaan jättämällä työstä pois muun muassa sivuston sisällön tuottaminen ja muokkaaminen. Jotta työstä ei olisi tullut liian laajaa, keskityttiin lähinnä rekistereiden ylläpitämiseen ja niiden sisällön esittämiseen. Yhdistykselle luotiin opinnäytetyössä sivustopohja, joka voidaan sellaisenaan ottaa käyttöön, mutta joka tarvitsee vielä tarkempaa paneutumista muun muassa yleisillä osioilla olevien tekstien muokkaamiseen ja päivittämiseen.

Yhdistyksen hallituksen ulkopuolisia jäseniä varten on luotu oma käyttäjäryhmänsä, jotta tulevaisuudessa voidaan sivustolle tehdä erityisesti heitä varten tarkoitettuja toimintoja. Lisäksi sivusto on suunniteltu silmälläpitäen mahdollisia laajennuksia esimerkiksi keskustelupalstaa, hallituksen jäsenten blogeja, tulostettavia raportteja rekistereistä yms. toimintoja.

Opinnäytetyössä ei käsitellä sivuston varsinaista käyttöönottoa, sillä työ on yksi osa kokonaisuutta Kainuun Kissanystävät ry:lle muodostettavasta uudesta sivustosta, lisäksi potentiaalisia palvelintarjoajia ei ole vielä kartoitettu. Sivustoa suunniteltaessa ja tehtäessä on huomioitu myös, että sivusto on muutettavissa käyttämään jotain muuta esimerkiksi MySQL-tietokantaa varsin helposti, mikäli sopivaa palvelintarjoajaa ei löydy tietokantojen osalta.

Jäsen- ja kissarekistereiden toimintaa ja käytettävyyttä on arvioitu yhteistyössä tyksen hallituksen jäsenten kanssa koko opinnäytetyöprosessin ajan. Rekisterit ja niiden hallintasivut ovat vastanneet yhdistyksen tarpeita ja vaatimuksia. Opinnäytetyönä laadittuun osaan yhdistyksen uudesta Internet-sivustosta tehdään mahdollisesti muutoksia vielä käyttöönoton jälkeenkin, sillä toiminnalliset osiot voidaan testata lopullisesti vasta oikeassa käytössä ja ympäristössä.

LÄHTEET

Evjen, B., Hanselman, S., Rader, D., Muhammad, F. & Sivakumar, S.Srinivasa. 2006. Professional ASP.NET 2.0 Special Edition. Indianapolis: Wiley Publishing, Inc.

Inkinen, V. 2003. ASP.NET. Jyväskylä: Docendo Finland Oy.

Khosravi, S. 2006. Professional ASP.NET 2.0 Server Control and Component Development. Indianapolis: Wiley Publishing, Inc.

Korpela, J. & Linjama, T. 2005. Web-suunnittelu. Jyväskylä: Docendo Finland Oy.

Kuutti, W. 2003. Käytettävyys, suunnittelu ja arviointi. Saarijärvi: Talentum Media Oy ja Wille Kuutti.

MSDN 2005 a. http://MSDN2.microsoft.com/en-us/library/aa479030.aspx (Luettu 4.9.2007)

MSDN b. http://MSDN2.microsoft.com/en-us/library/bwd43d0x(vs.80).aspx (Luettu 10.4.2007)

MSDN c. http://MSDN2.microsoft.com/en-us/library/ms227679(vs.80).aspx (Luettu 14.9.2007)

MSDN d. http://MSDN2.microsoft.com/en-us/library/2wawkw1c(vs.80).aspx (Luettu 20.9.2007)

MSDN e. http://MSDN2.microsoft.com/en-us/library/4w3ex9c2(VS.80).aspx (Luettu 20.9.2007)

Nielsen, J. 2000. WWW-suunnittelu. Jyväskylä: Oy Edita Ab

Rautiainen, T. Sihteeri. Kainuun Kissanystävät ry. Haastattelu 30.6.2007.

Wiio, A. 2004. Käyttäjäystävällisen sovelluksen suunnittelu. Helsinki: Edita Publishing Oy.

KIRJALLISUUS

Bellinaso, M. 2006. ASP.NET 2.0 Website Programming Problem – Design – Solution. Indianapolis: Wiley Publishing, Inc.

Hovi, A. 2004. SQL –opas. Jyväskylä: Docendo Finland Oy.

Hovi, A., Huotari, J. & Lahdenmäki, T. 2003. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo Finland Oy.

Järvinen, J. 2002. Hajautetut verkkopalvelut. Jyväskylä: Docendo Finland Oy.

Martin, J. & Tomson, B. 2002. ASP.NET Trainer Kit. Helsinki: Edita Publishing Oy.

Peltomäki, J. Inkinen, V. & Rantala, A. 2000. CGI-ja ASP-ohjelmointi. Jyväskylä: Teknolit Oy.

Platt, D.S. 2001. Microsoft .NET – Uudet ominaisuudet. Helsinki: Edita Oyj.

Salmela, J. 2002. Verkkosisällön hallinta. Helsinki: Edita Publishing Oy.

Wille, C.2001. C#. Jyväskylä: Docendo Finland Oy.

LIITTEET

Liite 1: Tiedostotyypit

Liite 2: Palvelindirektiivit

Liite 3: Tarkistuskontrollit

Tiedostotyypit (MSDN d.)

.asax	Tyypillisesti Global.asax on tiedosto, joka sisältää HttpApplication-luokasta johdettua koodia. Tiedostolla voidaan vaikuttaa sovelluksen elinkaaren alkamisen ja loppumisen toimintaan.
.ascx	Kehittäjän tekemien kontrollien tiedostopääte.
.ashx	Geneerinen käsittelijä, joka toteuttaa IHttpHandlerin liittymärajapinnan.
.asmx	XML-pohjainen Web-palvelu joka sisältää luokkia ja metodeita, jotka ovat toisten Web-sovellusten käytettävissä.
.aspx	ASP.NET sovelluksen sivu, joka sisältää palvelinpuolenkontrolleja.
.browser	Selaimen määrittely tiedosto. Tiedostojen avulla voidaan sivustoa kehittää myös selainkohtaisesti.
.config	Konfiguraatio tiedosto, yleensä Web.config. Sisältää XML-elementtejä joiden avulla esitetään sivuston asetukset.
.cs, .jsl, .vb	Sivustolla käytettävien luokkien lähdekoodi-tiedostot. Lisäksi ASP.NET sivujen CodeBehind-tiedostojen tiedostopäätteet.
.dll	Käännetty luokkakirjastotiedosto. Sijaitsee Bin-kansiossa.
.licx, .webinfo	Kontrollin lisenssitiedostot.
.master	MasterPage-tiedosto jolla voidaan määritellä sivustolle yhtenäinen rakenne.
.mdb, .ldb	Access-tietokanta tiedostot
.mdf	SQL Server Expressillä tehdyn tietokannan tietotyyppi.
.resources, .resx	Resurssitiedostot joiden avulla on mahdollista myös tehdä sivusto monelle eri kielelle.
.sitemap	Sivukartta-tiedosto jossa on kuvattu Web-sivuston rakenteen. Tiedostoa hyödynnetään erityisesti navigointikontrollien yhteydessä.
.skin	Tiedosto sisältää kontrollien ulkoasua koskevia määrittelyjä.

Palvelindirektiivit

@Page

AspCompat	Arvon ollessa true, sallii sivun suortittamisen yksi säikeisenä.
Async	Määrittelee käsitelläänkö sivua synkronisesti vai asynkronisesti.
AutoEventWireup	Määrittelee onko sivun tapahtumat toiminnassa
Buffer	Arvolla otetaan käyttöön HTTP vastauksen puskurointi
ClassName	Määrittelee sivun käyttämän luokan nimen
CodeFile	Viittaa CodeBehind-tiedostoon.
CodePage	Ilmaisee koodisivun arvon vastaukseen.
CompilerOptions	Merkkijono joka sisältää kääntäjän asetuksia
CompileWith	Merkkijonon arvo joka osoittaa käytettävään code-behind tiedostoon.
ContentType	Määrittelee HTTP vastauksen sisältötyypin.
Culture	Erittelee sivulla käytettävän kielen.
Debug	Ilmaisee sallitaanko debuggaus
Description	Tarjoaa sivun kuvauksen tekstinä.
EnableSessionState	Istunnon tilan päällelaitto
EnableTheming	Määrittelee sallitaanko sivun käyttää teemoja
EnableViewState	Määrittelee onko sivulla käytössä ViewState
ErroPage	Virhesivun polku kaikille käsittelemättömille virheille.
Language	Määrittelee käytettävän ohjelmointikielen
MasterPageFile	Merkkijono joka osoittaa käytettävään MasterPage-tiedostoon.
MaintainScrollPositionOn-	Määrittelee onko sivulla käytössä vierityspaikan palaut-

Postback	taminen sivun uudelleen lähettämisen jälkeen.
PersonalizationProvider	Merkkijono arvo jolla kohdistetaan sivulla käytettävään personointiprovideriin.
ResponseEncoding	Erittelee sivun sisällön koodaustavan
SmartNavigation	Määrittelee onko ASP.NETin Smart Navigation-ominaisuus päällä vai ei.
Src	Osoittaa sivun käyttämän luokan sijaintiin
Theme	Soveltaa tietyn teeman sivulle käyttäen ASP.NET 2.0 teema ominaisuuksia.
Title	Sivun otsikko
Trace	Asettaa sivun jäljityksen päälle, käytetään erityisesti kehi-tettäessä sivustoa

@Control

AutoEventWireup	Määrittelee onko sivun tapahtumat toiminnassa
ClassName	Määrittelee kontrollin käyttämän luokan nimen
CodeFile	Viittaa CodeBehind-tiedostoon.
CompilerOptions	Merkkijono joka sisältää kääntäjän asetuksia
CompileWith	Merkkijonon arvo joka osoittaa käytettävään code-behind tiedostoon.
Debug	Ilmaisee sallitaanko debuggaus
Description	Tarjoaa kontrollin kuvauksen tekstinä.
EnableTheming	Määrittelee sallitaanko sivun käyttää teemoja
EnableViewState	Määrittelee onko sivulla käytössä ViewState
Inherits	Määrittelee kontrollin käyttämän CodeBehind-luokan

Language	Määrittelee käytettävän ohjelmointikielen
Src	Osoittaa kontrollin käyttämän luokan sijaintiin

@Master

AutoEventWireup	Määrittelee onko MasterPagen tapahtumat toiminnassa
ClassName	Määrittelee MasterPagen käyttämän luokan nimen
CodeFile	Viittaa CodeBehind-tiedostoon.
CompilerOptions	Merkkijono joka sisältää kääntäjän asetuksia
CompileWith	Merkkijonon arvo joka osoittaa käytettävään code-behind tiedostoon.
Debug	Ilmaisee sallitaanko debuggaus
Description	Tarjoaa MasterPagen kuvauksen tekstinä.
EnableTheming	Määrittelee sallitaanko MasterPagen käyttää teemoja
EnableViewState	Määrittelee onko MasterPagen käytössä ViewState
Inherits	Määrittelee MasterPagen käyttämän CodeBehind-luokan
Language	Määrittelee käytettävän ohjelmointikielen
MasterPageFile	Merkkijono arvo jolla osoittaa MasterPagen käyttämään toiseen MasterPageen.
Src	Osoittaa kontrollin käyttämän luokan sijaintiin

@Import

Namespace	Sisällyttää arvona annetut nimiavaruudet sivun tai kontrolin käyttöön.

@Implements

Interface	Toteuttaa arvona annettavan .NET Frameworkin liittymärajapinnan, jolloin sivulla tai kontrollilla on pääsy kaikkiin sen metodeihin, tapahtumiin ja arvoihin

@ Register

Assembly	TagPrefixissa määritellyn aliaksen käyttämä kokoonpanotiedot.
Namespace	TagPrefixissa määritellyn aliaksen käyttämä nimiavaruus
Src	Kontrollin sijainti sovelluksessa
TagName	Aliasnimi käytettävälle luokannimelle
TagPrefix	Aliasnimi käytettävälle nimiavaruudelle

@ Assembly

Name	Mahdollistaa nimen antamisen, jonka avulla ilmaistaan kyseisen assemblyn käyttöä sivulla.
Src	Mahdollistaa tietyn lähteen määrittelemisen jota assembly-tiedosto käyttää käännökseen.

@ PreviousPageType

TypeName	Asettaa luokan nimen josta uudelleenlataus tulee.
VirtualPath	Asettaa sivun sijainnin josta uudelleenlataus tulee.

@ OutputCache

Duration	Ilmaisee ajan sekunteina jonka sivu tai kontrolli on välimuistissa.
SqlDependency	Ilmaisee tietokannan ja taulun nimen jonka tietoihin sivun välimuisti pohjautuu.
VaryByControl	Lista kontrolleista jotka sijoitetaan muistiin
VaryByHeader	Lista otsikkotiedoista jotka sijoitetaan muistiin
VaryByParam	Lista merkkijonoista joita käytetään tietojen välimuistiin sijoittamisessa

Tarkistuskontrollit (MSDN b.)

RequiredFieldValidator	Käyttäjän on täytettävä määritelty kenttä ennen kuin pystyy jatkamaan
CompareValidator	Vertaa käyttäjän antamaa syötettä joko vakioarvoon, toiseen kontrolliin tai tiettyyn tietotyyppiin. Vertailu voi olla pienempi-, suurempi- tai yhtäsuurikuin
RangeValidator	Tarkistaa käyttäjän antaman syötteen pituuden. Pituutta voi tarkistaa numeroilla, kirjaimilla ja päivämäärillä
RegularExpressionValidator	Tarkistaa käyttäjän antamaa syötettä vertaamalla sitä ennalta määriteltyihin arvoihin, joista muodostuu esimerkiksi postinumerot, sähköposti-osoite, puhelinnumeroihin jne.
CustomValidator	Tarkistaa käyttäjän antamaa syötettä sivustonkehittäjän tekemän määrittelyjen pohjalta
ValidationSummary	Kontrolli jonka avulla voidaan näyttää yhdessä paikassa virheilmoitukset joita edelliset kontrollit tuottavat